Exploring Controlled English OBDA

Camilo Thorne, Diego Calvanese

KRDB Centre

Free University of Bozen-Bolzano

Via della Mostra 4

39100 - Italy

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

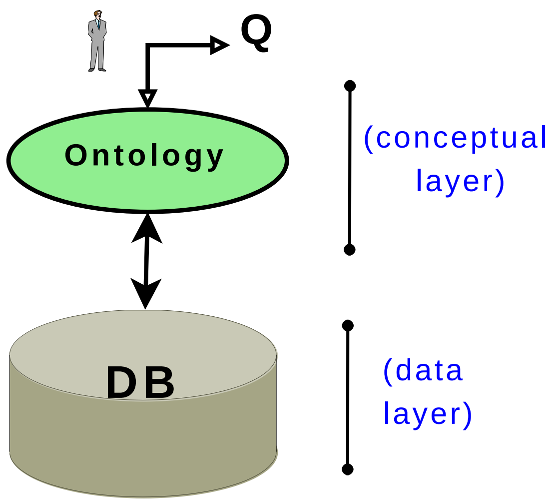
KRDB

Ontology-based systems [Staab&Studer 2004] aim at accessing and querying (possibly from the web) repositories of heterogenous 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)

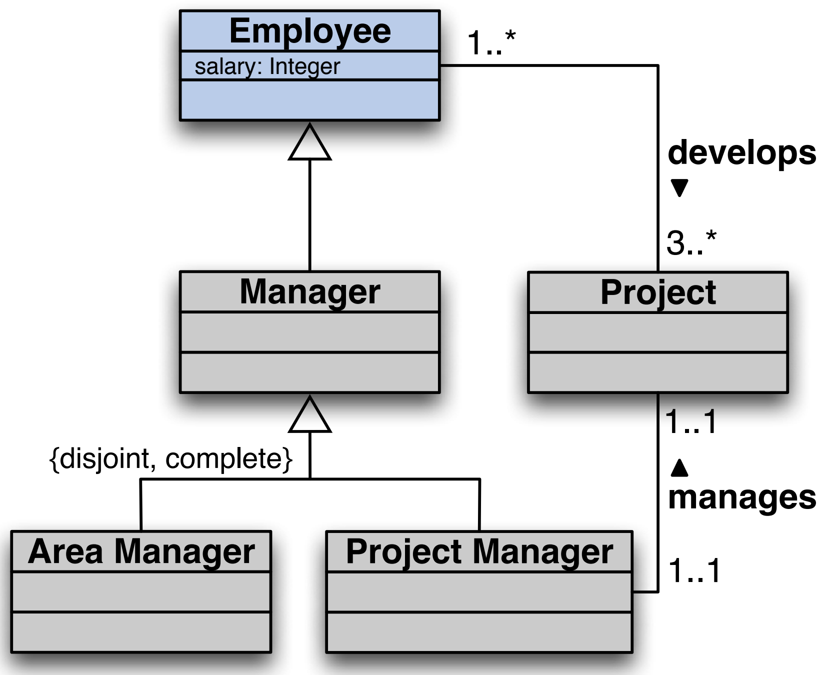
Ontology Languages

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

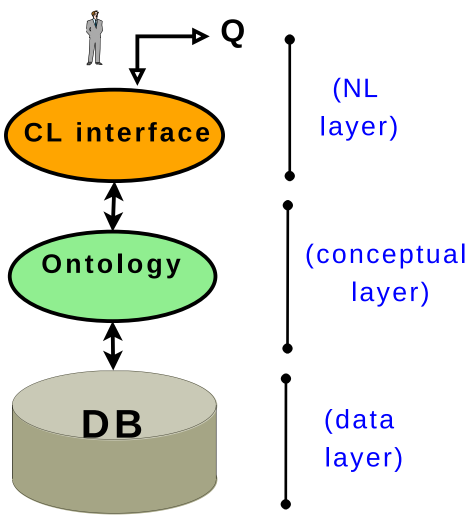
Controlled Languages

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]

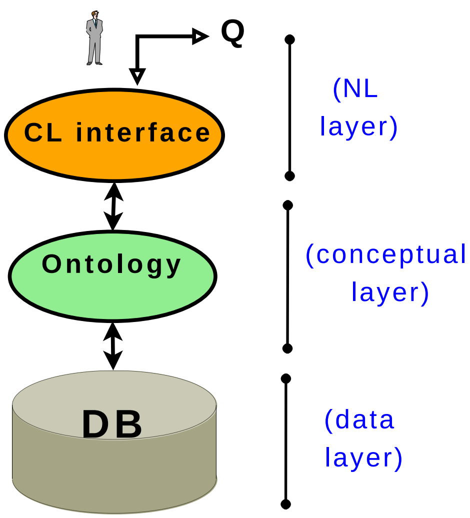


To improve the usability of interfaces to ontologies and OBDASs **controlled languages** [Bernstein et al. 2005, Sowa 2004] have been proposed

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

$\Rightarrow$ We should study the **computational complexity** of CLs w.r.t. OBDA

Semantic Web Language (OWL)

```
<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>
```

Description Logics + CL

$$Employee \sqsubseteq \exists develops{:}Project$$

Every employee develops some project

OWL is a **machine-readable** language (embedded in RDF and XML)

CLs are **human-readable**, yet as unambigous as DLs

Outline

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

1. OBDA and Query Answering
  (i) $\mathcal{ALCI}$ 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 $\{\text{IS-A}_i\}_{i \in [0,7]}$ fragments

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

4. Conclusions and further work

$\mathcal{ALCI}$ Ontologies

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

$$
\begin{array}{rcl}
R & \rightarrow & P \mid P^{-} \\
C & \rightarrow & \top \mid A \mid \exists R{:}C \mid \neg C \mid C \sqcap C'
\end{array}
$$

$\mathcal{ALCI}$ Ontologies

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

$$\begin{array}{rcl} R & \to & P \mid P^- \\ C & \to & \top \mid A \mid \exists R{:}C \mid \neg C \mid C \sqcap C' \end{array}$$

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)

$\mathcal{ALCI}$ Ontologies

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

$$\begin{array}{rcl} R & \rightarrow & P \mid P^{-} \\ C & \rightarrow & \top \mid A \mid \exists R{:}C \mid \neg C \mid C \sqcap C' \end{array}$$

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** $\mathcal{D} := \langle \Delta, \cdot^{\mathcal{D}} \rangle$

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

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

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

$Mod(\langle \mathcal{T}, \mathcal{A} \rangle) := \{\mathcal{D} \mid \mathcal{D} \models \langle \mathcal{T}, \mathcal{A} \rangle\}$

Conjunctive 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**

Conjunctive 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

Conjunctive Queries

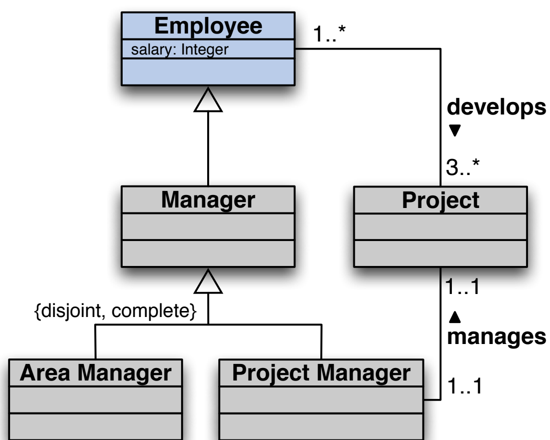
A **conjunctive query** (CQ) is a query of the form

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

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?

$$\begin{aligned} q(x) \quad \leftarrow \quad & Manager(x) \wedge ProjectManager(x) \\ & \wedge \exists y(manages(x,y) \wedge Project(y)) \end{aligned}$$

```
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 $\langle \mathcal{T}, \mathcal{A} \rangle$ are:

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

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

$\Rightarrow$ 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 $\langle \mathcal{T}, \mathcal{A} \rangle$ are:

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

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

$\Rightarrow$ asking q to an ontology = 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

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:

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

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

$\Rightarrow$ asking q to an ontology = 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 *DL-Lite*

In *DL-Lite* concepts are partitioned into **right** and **left** concepts:

$$\begin{array}{rcl} R & \to & P \mid P^{-} \\ C_l & \to & A \mid \exists R{:}\top \\ C_r & \to & C_l \mid \neg C_l \mid C_r \sqcap C_r' \mid \exists R{:}C_r \end{array}$$

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

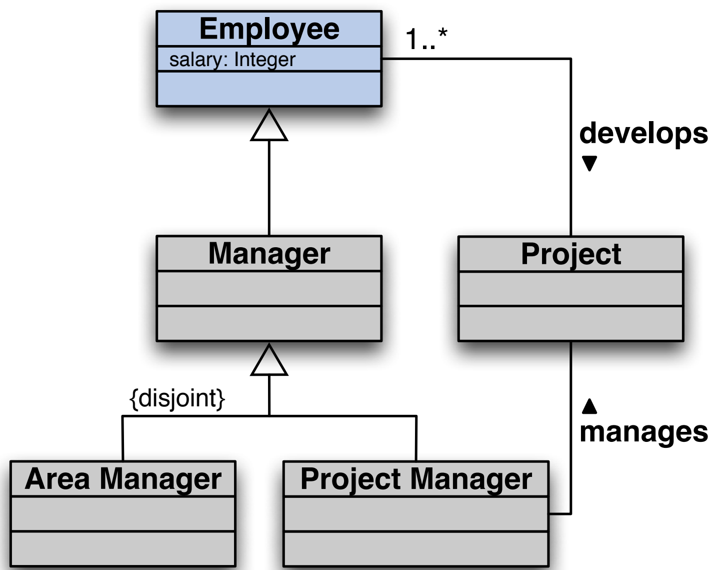
QA (w.r.t. CQs) is optimal $\Rightarrow$ **LogSpace** in data complexity

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

$\Rightarrow$ *DL-Lite* **scales to data**!

DL-Lite Ontologies

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



$$
\begin{aligned}
\mathit{Manager} &\sqsubseteq \mathit{Employee}\\
\mathit{AreaManager} &\sqsubseteq \mathit{Manager}\\
\mathit{ProjectManager} &\sqsubseteq \mathit{Manager}\\
\mathit{AreaManager} &\sqsubseteq \neg \mathit{ProjectManager}\\[1ex]
\exists \mathit{develops} &\sqsubseteq \mathit{Employee}\\
\exists \mathit{develops}^- &\sqsubseteq \mathit{Project}\\
\mathit{Project} &\sqsubseteq \exists \mathit{develops}^-\\[1ex]
\exists \mathit{manages} &\sqsubseteq \mathit{TopManager}\\
\exists \mathit{manages}^- &\sqsubseteq \mathit{Project}\\
\mathit{TopManager} &\sqsubseteq \exists \mathit{manages}\\
\mathit{Project} &\sqsubseteq \exists \mathit{manages}^-\\[1ex]
\mathit{Employee} &\sqsubseteq \exists \mathit{has}{:}\mathit{Salary}
\end{aligned}
$$

NB: in *DL-Lite* we cannot capture completeness of the hierarchy

Expressing $\mathcal{ALCI}$ ontologies with DL-English

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

$\Rightarrow$ how difficult is it to access data from an ontology with CL?

$\Rightarrow$ does this task **scale to data**?

Expressing $\mathcal{ALCI}$ ontologies with DL-English

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

- word categories **N**, **Adj** and **IV** to atomic concepts
- category **TV** to role names
- recursive constituents to arbitrary concepts

$$\overbrace{\lambda C.\lambda C'.C \sqsubseteq C'}^{\textbf{Every}} \quad \overbrace{C}^{\textbf{Nom}} \quad \overbrace{C'}^{\textbf{VP}} \qquad\qquad \overbrace{\lambda C.\lambda C'.C \sqsubseteq C'}^{\textbf{Everybody who}} \quad \overbrace{C}^{\textbf{VP}} \quad \overbrace{C'}^{\textbf{VP}}$$

No manager who manages some project that does not make some money is shrewd.
$Manager \sqcap \exists manages{:}(Project \sqcap \neg(\exists make{:}Money)) \sqsubseteq \neg Shrewd$

Nobody manages only projects
$\forall manages{:}Project \sqsubseteq \bot$

Anybody who manages some project manages some big project or small project
$\exists manages{:}Project \sqsubseteq \exists manages{:}((Project \sqcap Big) \sqcup ((Project \sqcap Small)$

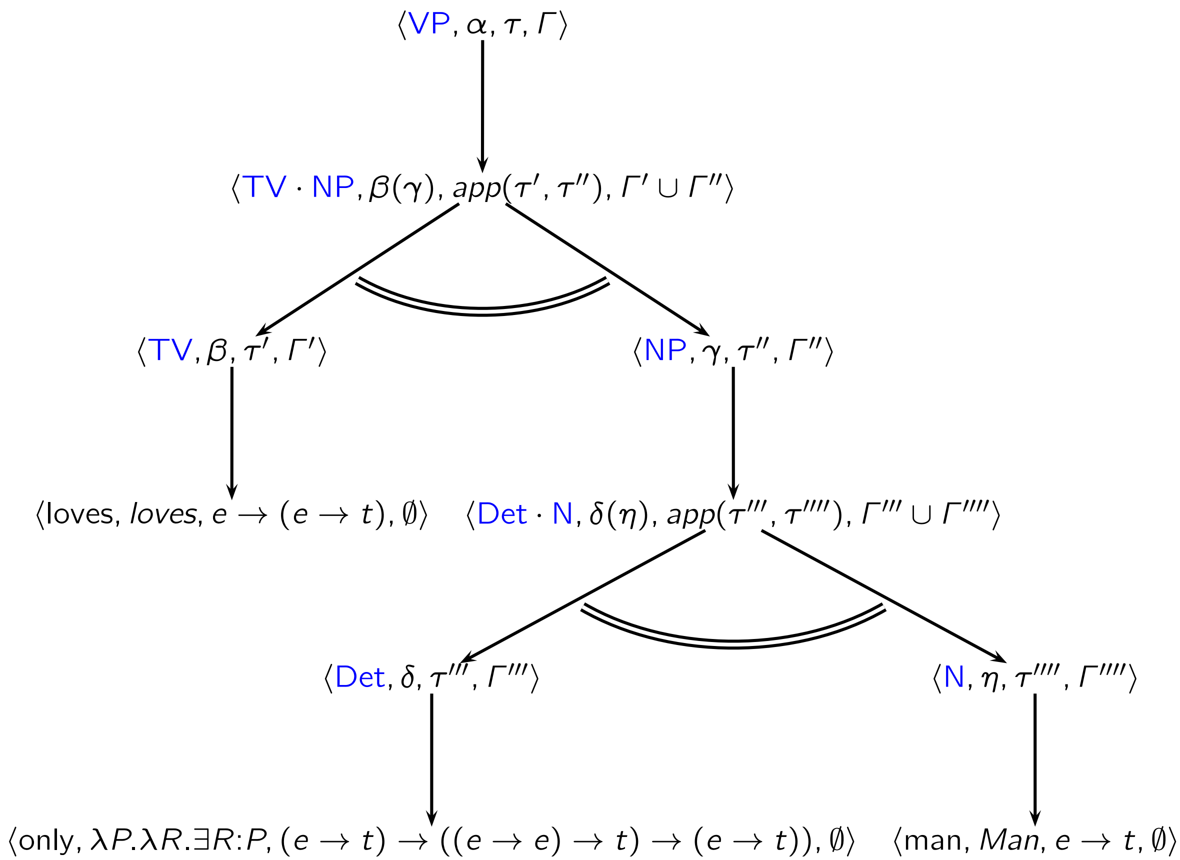
All DL-English (complete) sentences translate into an $\mathcal{ALCI}$ assertion and conversely

Expressing $\mathcal{ALCI}$ ontologies with DL-English

S $\rightarrow$ NP VP			NP $\rightarrow$ Det Nom
VP $\rightarrow$ TV NP	VP $\rightarrow$ is a Nom	VP $\rightarrow$ is TV by NP	NP $\rightarrow$ Pro Relp VP
VP $\rightarrow$ is Adj	VP $\rightarrow$ IV	VP $\rightarrow$ is Neg TV by NP	NP $\rightarrow$ Pro
VP $\rightarrow$ does Neg IV	VP $\rightarrow$ is Neg a Nom	Nom $\rightarrow$ Nom Relp VP	Nom $\rightarrow$ Adj Nom
VP $\rightarrow$ is Neg Adj	VP $\rightarrow$ VP Crd VP	Nom $\rightarrow$ Nom Crd Nom	Nom $\rightarrow$ N

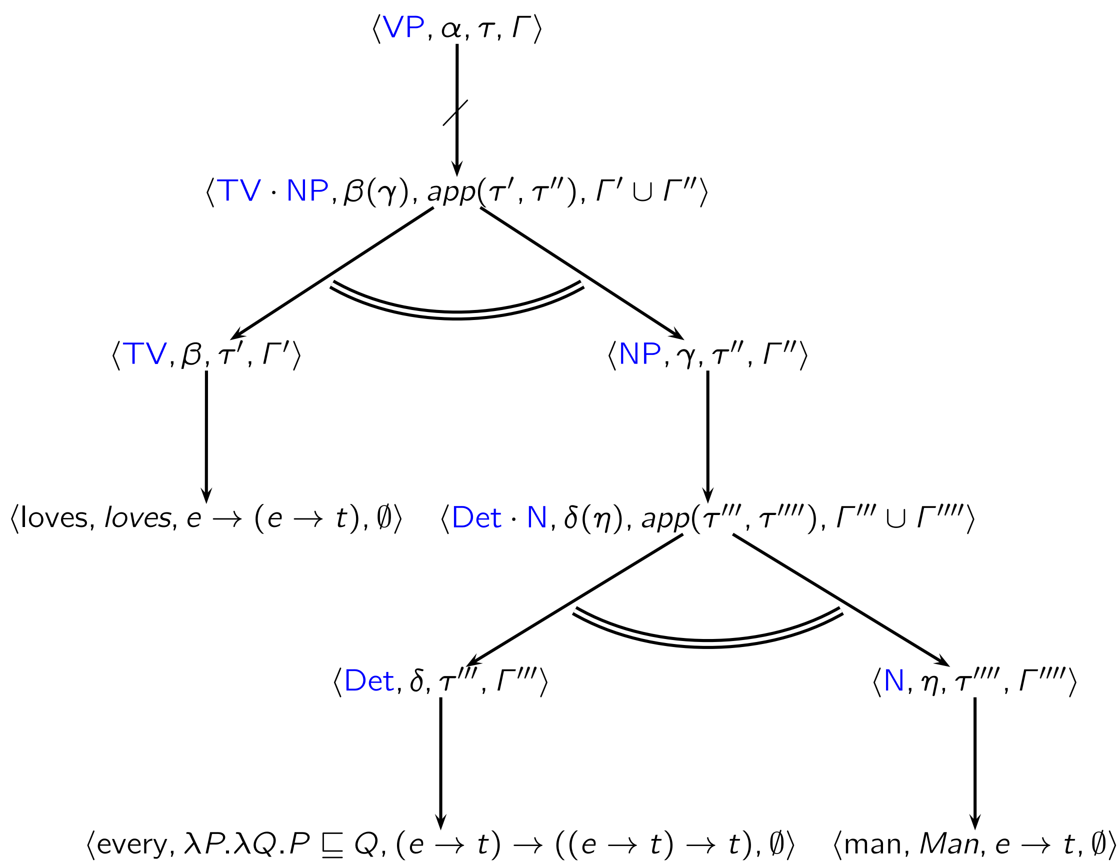
$\tau(\text{VP}) := \tau(\text{NP})(\tau(\text{TV}))$	$\tau(\text{VP}) := \tau(\text{Crd})(\tau(\text{VP}))(\tau(\text{VP}))$	$\tau(\text{S}) := \tau(\text{NP})(\tau(\text{VP}))$
$\tau(\text{VP}) := \tau(\text{Neg})(\tau(\text{NP})(\tau(\text{TV})))$	$\tau(\text{VP}) := \tau(\text{Neg})(\tau(\text{Adj}))$	
$\tau(\text{VP}) := \tau(\text{Neg})(\tau(\text{Nom}))$	$\tau(\text{VP}) := \tau(\text{Neg})(\tau(\text{IV}))$	$\tau(\text{VP}) := \tau(\text{Adj})$
$\tau(\text{NP}) := \tau(\text{Pro})$	$\tau(\text{VP}) := \tau(\text{IV})$	$\tau(\text{VP}) := \tau(\text{Nom})$
$\tau(\text{NP}) := \tau(\text{Det})(\tau(\text{Nom}))$	$\tau(\text{NP}) := \tau(\text{Pro})(\tau(\text{Relp})(\tau(\text{VP})))$	$\tau(\text{Nom}) := \tau(\text{N})$
$\tau(\text{Nom}) := \tau(\text{Nom})(\tau(\text{Relp})(\tau(\text{VP})))$	$\tau(\text{Nom}) := \tau(\text{Crd})(\tau(\text{Nom}))(\tau(\text{Nom}))$	$\tau(\text{Nom}) := \tau(\text{Adj})(\tau(\text{Nom}))$

Pro $\rightarrow$ anybody	$\tau(\text{Pro}) := \lambda C.\lambda C'.C \sqsubseteq C'$:	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$
Pro $\rightarrow$ somebody	$\tau(\text{Pro}) := \lambda R.\exists R$:	$(e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$
Pro $\rightarrow$ nobody	$\tau(\text{Pro}) := \lambda C.\lambda C'.C \sqsubseteq \neg C'$:	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$
Pro $\rightarrow$ nobody	$\tau(\text{Pro}) := \lambda R.\neg\exists R$:	$(e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$
Crd $\rightarrow$ and	$\tau(\text{Crd}) := \lambda C.\lambda C'.C \sqcap C'$:	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$
Crd $\rightarrow$ or	$\tau(\text{Crd}) := \lambda C.\lambda C'.C \sqcup C'$:	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$
Relp $\rightarrow$ who	$\tau(\text{Relp}) := \lambda C.C$:	$(e \rightarrow t) \rightarrow (e \rightarrow t)$
Neg $\rightarrow$ not	$\tau(\text{Neg}) := \lambda C.\neg C$:	$(e \rightarrow t) \rightarrow (e \rightarrow t)$
Pro $\rightarrow$ only	$\tau(\text{Pro}) := \lambda C.\lambda R.\forall R{:}C$:	$(e \rightarrow t) \rightarrow ((e \rightarrow (e \rightarrow t) \rightarrow (e \rightarrow t)$
Pro $\rightarrow$ everybody	$\tau(\text{Pro}) := \lambda C.\top \sqsubseteq C$:	$(e \rightarrow t) \rightarrow t$
Pro $\rightarrow$ nobody	$\tau(\text{Pro}) := \lambda C.C \sqsubseteq \bot$:	$(e \rightarrow t) \rightarrow t$
Det $\rightarrow$ some	$\tau(\text{Det}) := \lambda C.\lambda R.\exists R{:}C$:	$(e \rightarrow t) \rightarrow ((e \rightarrow (e \rightarrow t) \rightarrow (e \rightarrow t)$
Det $\rightarrow$ every	$\tau(\text{Det}) := \lambda C.\lambda C'.C \sqsubseteq C'$:	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$
Det $\rightarrow$ no	$\tau(\text{Det}) := \lambda C.\lambda C'.C \sqsubseteq \neg C'$:	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$



A succesful derivation for the **VP** "loves only men"
$\Rightarrow$ types unify

Expressing $\mathcal{ALCI}$ ontologies with DL-English

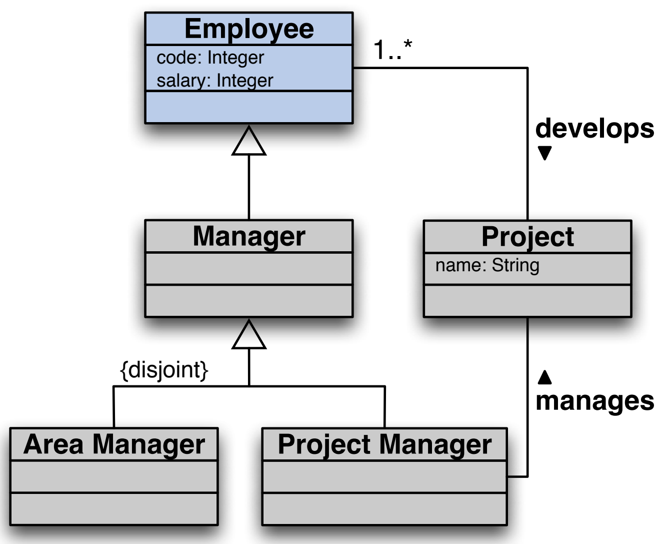


Failed derivation for the **VP** "loves every man"
$\Rightarrow$ types $e \to (e \to t)$ and $e \to t$ do not unify.

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"



Every area manager is a manager
Every project manager is a manager
Every manager is an employee
No project manager is an area manager

Anybody who develops something is an employee
Anything that is developed by somebody is a project

Anybody who manages something is a project manager
Anything that is managed by somebody is a project

Every project is developed by some employee

Every employee has some salary
Every employee has some code
Every project has some name

DL-Lite is expressed by Lite-English [Bernardi et al. 2007]

Related Declarative CLs

CL (English)	Maps to	Goal
ACE [Fuchs 2005]	FO	KR/User specifications
ACE-OWL [Kaaljurand 2007]	OWL-DL	Ontology authoring + querying
PENG [Schwitter 2003]	OWL-DL	Ontology authoring + querying
SOS [Schwitter2008]	OWL-DL	Ontology authoring + querying
CLCE [Sowa2004]	FOL	Knowledge representation
AECMA [Unwalla 2005]	no	User specifications
English Query (EQ) [Blum 1999]	SQL	DB querying/management
OWL-CNL [Schwitter 2006]	OWL-DL	Ontology authoring
Easy English [Bernth 1998]	no	User specifications
λ-SQL [Winter 2006]	SQL	DB querying
nRQL [Schwitter 2008]	FO queries	Ontology querying
Rabbit [Schwitter2008]	OWL	Ontology authoring
ACE-PQL [Bernstein 2005]	PQL	Ontology querying
QE-III [Clifford 1987]	IL	DB querying

(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 λ-abstraction, λ-application, types and β-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 λ-abstraction, λ-application, types and β-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_{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

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.)

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,

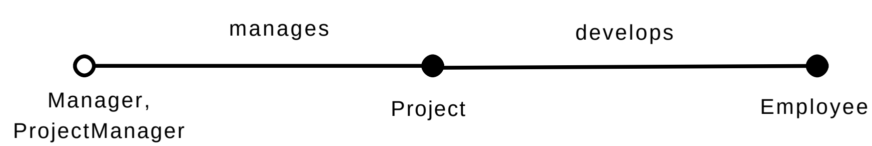
- the determiner "some" and the pronouns "something, somebody" (existential)
- relative pronouns and **VP**-coordination (conjunction)
- interrogative pronouns such as "which, what, who," (etc.)

EXAMLE:

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

$$q(x) \leftarrow Manager(x) \wedge ProjectManager(x) \wedge \exists y(manages(x,y) \wedge Project(y)$$
$$\exists z(develops(z,y) \wedge Employee(z)) \wedge Project(y)))$$

$$\lambda x^e.Manager(x) \wedge ProjectManager(x) \wedge \exists y(manages(x,y) \wedge Project(y)$$
$$\exists z(develops(z,y) \wedge Employee(z)) \wedge Project(y))) : e \rightarrow t$$



The Family $\{\text{IS-A}_i\}_{i \in [0,7]}$ 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 $\{\text{IS-A}_i\}_{i \in [0,7]}$ 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

We adopt as strategy **restricting** on DL-English

Utterances in each fragment translate into assertions $C_l \sqsubseteq C_r$

Hence, we partition [Bernardi et al. 2007] **Nom** and **VP** into

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

$\Rightarrow$ this allows for a fine-grained data complexity analysis

The Family $\{\text{IS-A}_i\}_{i\in[0,7]}$ of CLs

Fragment	Assertions		Sample Sentence(s)
IS-A$_0$	$A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	$\Rightarrow$	Every project manager is a manager and is an employee.
IS-A$_1$	$A \sqsubseteq \forall P{:}A'$	$\Rightarrow$	Every project manager manages only projects.
IS-A$_2$	$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq \forall P{:}(A_1 \sqcap \cdots \sqcap A_m)$	$\Rightarrow$	Every good manager manages only good projects.
IS-A$_3$	$\exists P{:}A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	$\Rightarrow$	Anybody who manages some project is an employee and is a manager.
	$\exists P^{-}{:}A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	$\Rightarrow$	Anything that is managed by some important manager is a big project.
	$A \sqsubseteq \exists P$	$\Rightarrow$	Every manager manages something.
IS-A$_4$	$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A_1 \sqcap \cdots \sqcap A_m$	$\Rightarrow$	Every cruel manager is a bad manager.
	$\exists P{:}(A_1 \sqcap \cdots \sqcap A_n) \sqsubseteq A_1 \sqcap \cdots \sqcap A_m$	$\Rightarrow$	Anybody who manages some bankrupt project is a bad manager.
IS-A$_5$	$\forall P{:}A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	$\Rightarrow$	Anybody who manages only projects is a manager and a project manager.
IS-A$_6$	$A \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$	$\Rightarrow$	Every manager is a project manager or is an area manager.
IS-A$_7$	$\neg A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	$\Rightarrow$	Anybody who is not an area manager is an employee who is a project manager.

The Family $\{\text{IS-A}_i\}_{i\in[0,7]}$ of CLs

$$\underbrace{\lambda C_l.\lambda C_r.C_l \sqsubseteq C_r}_{\textbf{every}} \quad \underbrace{C_l}_{\textbf{Nom}_r} \quad \underbrace{C_r}_{\textbf{VP}_l} \qquad\qquad \underbrace{\lambda C_l.\lambda C_r.C_l \sqsubseteq C_r}_{\textbf{everybody who}} \quad \underbrace{C_l}_{\textbf{VP}_l} \quad \underbrace{C_r}_{\textbf{VP}_r}$$

Concept C_f	Constituent α_f	Grammar Rules
A $\exists P : A$ $\exists P^- : A$ $\forall P : A$	Nom$_f$, VP$_f$ TV some Nom$_f$, TV somebody who VP$_f$ TV by some Nom$_f$, TV by somebody who VP$_f$ TV only VP$_f$, TV only who VP$_f$	VP$_f$ $\to$ is a Nom$_f$ | IV | is Adj Nom$_f$ $\to$ N
$\exists P$	TV something, TV somebody	$\emptyset$
$A_1 \sqcap \cdots \sqcap A_n$	Adj Nom$_f$, Nom$_f$ who VP$_f$	VP$_f$ $\to$ is a Nom$_f$ | IV | is Adj | VP$_f$ and VP$_f$
	Nom$_f$ and Nom$_f$, VP$_f$ and VP$_f$	Nom$_f$ $\to$ N | Adj Nom$_f$ | Nom$_f$ and Nom$_f$
$A_1 \sqcup \cdots \sqcup A_n$	VP$_f$ or VP$_f$	VP$_f$ $\to$ is a Nom$_f$ | IV | is Adj | VP$_f$ and VP$_f$ Nom$_f$ $\to$ N | Nom$_f$ and Nom$_f$
$\neg A$	is not Adj, does not IV, is not a Nom$_f$	Nom$_f$ $\to$ N

Complexity (w.r.t. TCQs)

	SAT (KB)	QA (data)	QA (combined)
IS-A_0	in **PTime**	in **LogSpace**	in **PTime**
IS-A_1	in **PTime**	**NLogSpace**-complete	in **PSpace**
IS-A_2	in **PTime**	**PTime**-complete	in **PSpace**
IS-A_3	in **PTime**	**PTime**-complete	in **NExpTime** (*)
IS-A_4	in **PTime**	**PTime**-complete	in **PSpace**
IS-A_5	in **PTime**	**coNP**-complete	in **NExpTime** (*)
IS-A_6	in **PTime**	**coNP**-complete	**coNP**-complete
IS-A_7	in **PTime**	**coNP**-complete	**coNP**-complete

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)

	SAT (**KB**)	**QA** (**data**)	**QA** (**combined**)
IS-A_0	in **PTime**	in **LogSpace**	in **PTime**
IS-A_1	in **PTime**	**NLogSpace**-complete	in **PSpace**
IS-A_2	in **PTime**	**PTime**-complete	in **PSpace**
IS-A_3	in **PTime**	**PTime**-complete	in **NExpTime** (*)
IS-A_4	in **PTime**	**PTime**-complete	in **PSpace**
IS-A_5	in **PTime**	**coNP**-complete	in **NExpTime** (*)
IS-A_6	in **PTime**	**coNP**-complete	**coNP**-complete
IS-A_7	in **PTime**	**coNP**-complete	**coNP**-complete

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 $\mathcal{ELI}$ $\quad C \to \top \mid A \mid \exists R{:}C \mid C \sqcap C'$

⇒ medical ontologies (e.g. GALEN) express mostly $\mathcal{ELI}$ assertions

Complexity (w.r.t. TCQs)

	SAT (KB)	**QA (data)**	**QA (combined)**
DL-Lite	in **PTime**	in **LogSpace**	in **PSpace** (*)
$\mathcal{ALCI}$	**ExpTime**-complete	**coNP**-complete	**ExpTime**-complete
$\mathcal{ALCQI}$	**ExpTime**-complete	**coNP**-complete	**ExpTime**-complete
$\mathcal{SHIF}$	**ExpTime**-complete	**coNP**-complete	**ExpTime**-complete
$\mathcal{SHOIN}$	**NExpTime**-complete	**coNP**-hard	**NExpTime**-complete
$\mathcal{SHROIQ}$	**NExpTime**-hard	**coNP**-hard	**NExpTime**-hard

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

DL-Lite	=	Lite-English		
$\mathcal{ALCI}$	=	DL-English		
$\mathcal{SHIF}[\mathcal{D}]$	=	ACE-OWL-Lite	=	OWL-Lite
$\mathcal{SHOIN}[\mathcal{D}]$	=	ACE-OWL-DL	=	OWL-DL
$\mathcal{SROIQ}[\mathcal{D}]$	=	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 $\mathcal{ALCI}$ 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 **aggregration functions** does not (over *DL-Lite* ontologies)