Ontology Based Data Access

Vision: Ontologies at the Core of Information Systems

- Usage of all system resources (data and services) is done through a domain conceptualization.
- Cooperation between systems is done at the level of the conceptualizations.
- This implies:
  - Hide to the user where and how data and services are stored or implemented;
  - Present to the user a conceptual view of the data and services.

Ontology based Data Access

- An ontology provides meta-information about the data and the vocabulary used to query the data. It can impose constraints on the data.
- Actual data can be incomplete w.r.t. such meta-information and constraints. So data should be stored using open world semantics rather than closed world semantics: use ABoxes instead of relational database instances.
- During query answering, the system has to take into account the ontology.

We discuss ontology based data access in the framework of description logic knowledge bases.

Knowledge Base (KB) =TBox+ABox

TBox (terminological box, schema)

Man $\equiv$ Human $\sqcap$ Male
Father $\equiv$ Man $\sqcap$ $\exists$hasChild
...

ABox (assertion box, data)

john : Man
(john, mary) : hasChild
...

Inference System

Interface

Knowledge Base (= Ontology with database instance)

A **knowledge base** $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and a simple ABox $\mathcal{A}$ (or, equivalently, a database instance).

We combine the open world semantics for TBoxes and ABoxes in the obvious manner, and obtain an **open world semantics** for knowledge bases.

An interpretation $\mathcal{I}$ **satisfies** a knowledge base $(\mathcal{T}, \mathcal{A})$, in symbols

$I \models T \wedge I \models A$

$$\mathcal{I} \models (\mathcal{T}, \mathcal{A}),$$

if it satisfies both $\mathcal{T}$ and $\mathcal{A}$. In this case we also say that $\mathcal{I}$ is a **model** of $(\mathcal{T}, \mathcal{A})$. The set of models of $(\mathcal{T}, \mathcal{A})$ is denoted by $\mathbf{Mod}(\mathcal{T}, \mathcal{A})$.

Certain Answers

Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and an FOPL query $F(x_1, \dots, x_k)$, we say that $(a_1, \dots, a_k)$ is a **certain answer** to $F(x_1, \dots, x_k)$ by $\mathcal{K}$, in symbols

$$\mathcal{K} \models F(a_1, \dots, a_k),$$

if

- $a_1, \dots, a_k$ are individual names in $\mathcal{A}$;
- for all interpretations $\mathcal{I}$:

$$\mathcal{I} \models \mathcal{K} \quad \Rightarrow \quad \mathcal{I} \models F(a_1, \dots, a_k).$$

The set of certain answers given to F by $\mathcal{K}$ is defined as:

$$\textbf{certanswer}(F, \mathcal{K}) = \{(a_1, \dots, a_k) \mid \mathcal{K} \models F(a_1, \dots, a_k)\}$$

Boolean Queries

Let $\mathcal{K}$ be a knowledge base. For a query $\boldsymbol{F}$ without variables (Boolean query), we say that

- the certain answer given by $\mathcal{K}$ is "yes" if $\mathcal{I} \models \boldsymbol{F}$, for all interpretations $\mathcal{I}$ satisfying $\mathcal{K}$;
- the certain answer given by $\mathcal{K}$ is "no" if $\mathcal{I} \not\models \boldsymbol{F}$, for all interpretations $\mathcal{I}$ satisfying $\mathcal{K}$.
- Otherwise the certain answer is: "Don't know".

Example

Consider the TBox $\mathcal{T}_U$:

- $\textbf{BritishUniversity} \sqsubseteq \textbf{University}$;
- $\textbf{University} \sqcap \textbf{Student} \sqsubseteq \bot$;
- $\top \sqsubseteq \forall \textbf{registered\_at}.\textbf{University}$;
- $\top \sqsubseteq \forall \textbf{student\_at}.\textbf{University}$;
- $\exists \textbf{student\_at}.\top \sqsubseteq \textbf{Student}$;
- $\textbf{Student} \sqsubseteq \exists \textbf{student\_at}.\top$;
- $\textbf{NonBritishUni} \equiv \textbf{University} \sqcap \neg \textbf{BritishUniversity}$.

Example (continued)

and the simple ABox (equivalently, database instance) $\mathcal{A}$:

- $\textbf{NonBritishUni}(\textbf{CMU})$
- $\textbf{Institution}(\textbf{Harvard}), \quad \textbf{Institution}(\textbf{FUBerlin})$
- $\textbf{BritishUniversity}(\textbf{LU}), \quad \textbf{BritishUniversity}(\textbf{MU})$
- $\textbf{Student}(\textbf{Tim})$
- $\textbf{registered}(\textbf{Tim}, \textbf{LU}), \quad \textbf{registered}(\textbf{Bob}, \textbf{MU})$
- $\textbf{student\_at}(\textbf{Tom}, \textbf{Harvard})$

Example (continued)

Denote by $\mathcal{I}_{\mathcal{A}}$ the interpretation corresponding to the database instance $\mathcal{A}$:

- $\Delta^{\mathcal{I}_{\mathcal{A}}} = \{\mathbf{CMU}, \mathbf{Harvard}, \mathbf{FUBerlin}, \mathbf{Tim}, \mathbf{Tom}, \mathbf{Bob}, \mathbf{MU}, \mathbf{LU}\}$;
- $\mathbf{NonBritishUni}^{\mathcal{I}_{\mathcal{A}}} = \{\mathbf{CMU}\}$;
- $\mathbf{Institution}^{\mathcal{I}_{\mathcal{A}}} = \{\mathbf{Harvard}, \mathbf{FUBerlin}\}$;
- $\mathbf{BritishUniversity}^{\mathcal{I}_{\mathcal{A}}} = \{\mathbf{LU}, \mathbf{MU}\}$;
- $\mathbf{Student}^{\mathcal{I}_{\mathcal{A}}} = \{\mathbf{Tim}\}$;
- $\mathbf{registered\_at}^{\mathcal{I}_{\mathcal{A}}} = \{(\mathbf{Tim}, \mathbf{LU}), (\mathbf{Bob}, \mathbf{MU})\}$;
- $\mathbf{student\_at}^{\mathcal{I}_{\mathcal{A}}} = \{(\mathbf{Tom}, \mathbf{Harvard})\}$.

(Certain) Answers

In the table below, we consider Boolean queries $C(a)$ (in description logic notation!) and give the (certain) answer to $C(a)$ of the database instance $\mathcal{I}_{\mathcal{A}}$, the ABox $\mathcal{A}$, and the knowledge base $\mathcal{K}_U = (\mathcal{T}_U, \mathcal{A})$.

CMU是不是University

Boolean Query	$\mathcal{I}_{\mathcal{A}}$	Abox $\mathcal{A}$	KB $\mathcal{K}_U$
University(**CMU**)	No	Don't know	Yes
University(**Harvard**)	No	Don't know	Yes
NonBritishUni(**CMU**)	Yes	Yes	Yes
Student(**Tim**)	Yes	Yes	Yes
Student(**Tom**)	No	Don't know	Yes
$\exists$**student_at**.$\top$(**Tom**)	Yes	Yes	Yes
$\exists$**student_at**.$\top$(**Tim**)	No	Don't know	Yes
(**Student** $\sqcap$ $\neg$**University**)(**Tim**)	Yes	Don't know	Yes
(**Institution** $\sqcap$ $\neg$**University**)(**FUBerlin**)	Yes	Don't know	Don't know

I_A: closed world assumption

ABox A: open world

Example

Let $\mathcal{S} = (\mathcal{O}, \mathcal{B})$ be a knowledge base with simple ABox $\mathcal{B}$ given by

$$\textbf{Person}(\textbf{john}), \quad \textbf{Person}(\textbf{nick}), \quad \textbf{Person}(\textbf{toni})$$

$$\textbf{hasFather}(\textbf{john}, \textbf{nick}), \quad \textbf{hasFather}(\textbf{nick}, \textbf{toni})$$

and TBox $\mathcal{O}$ defined as

$$\mathcal{O} = \{\textbf{Person} \sqsubseteq \exists \textbf{has\_Father}.\textbf{Person}\}$$

For the FOPL query

$$F(x, y) = \textbf{hasFather}(x, y)$$

we obtain

$$\textbf{certanswer}(F, \mathcal{S}) = \{(\textbf{john}, \textbf{nick}), (\textbf{nick}, \textbf{toni})\}.$$

Example

- For the query

$$F(x) = \exists y.\mathsf{hasFather}(x, y)$$

  we obtain

$$\mathsf{certanswer}(F(x), \mathcal{S}) = \{\mathsf{john}, \mathsf{nick}, \mathsf{toni}\}$$

- For the query

$$F(x) = \exists y_1 \exists y_2 \exists y_3.(\mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3))$$

  we obtain

$$\mathsf{certanswer}(F(x), \mathcal{S}) = \{\mathsf{john}, \mathsf{nick}, \mathsf{toni}\}$$

- For the query

$$F(x, y_3) = \exists y_1 \exists y_2.(\mathsf{hasFather}(x, y_1) \wedge \mathsf{hasFather}(y_1, y_2) \wedge \mathsf{hasFather}(y_2, y_3))$$

  we obtain

$$\mathsf{certanswer}(F(x, y_3), \mathcal{S}) = \emptyset$$

Complexity of querying $(\mathcal{T}, \mathcal{A})$

Consider, for simplicity, Boolean queries. There are two different ways of measuring the complexity of querying:

- Data complexity: Measures the time/space needed to evaluate a fixed query $\boldsymbol{F}$ for a fixed TBox $\boldsymbol{\mathcal{T}}$ in $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{A}})$ (i.e., check $\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{A}}) \models \boldsymbol{F}$). The only input variable is the size of $\boldsymbol{\mathcal{A}}$.

- Combined complexity: Measure the time/space needed to evaluate a query in $(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{A}})$. The input variables are the size of the query, the size of $\boldsymbol{\mathcal{T}}$, and the size of $\boldsymbol{\mathcal{A}}$.

Data complexity is relevant if $\boldsymbol{\mathcal{T}}$ and the query are very small compared to $\boldsymbol{\mathcal{A}}$. This is the case in most applications.

Non-Tractability of Query answering in $\mathcal{ALC}$ in Data Complexity

A graph G is a pair (W, E) consisting of a set W and a symmetric relation E on W.

G is 3-colorable if there exist subsets **blue**, **red**, and **green** of W such that

- the sets **blue**, **green**, and **red** are mutually disjoint;
- **blue** $\cup$ **red** $\cup$ **green** $= W$;
- if $(a, b) \in E$, then a and b do not have the same color.

3-colorability of graphs is an NP-complete problem.

3-Colorability as a Query Answering Problem

Assume $G = (W, E)$ is given. Construct the ABox $\mathcal{A}_G$ by taking a role name r and setting

- $r(a, b) \in \mathcal{A}$ for all $a, b \in W$ with $(a, b) \in E$.

Construct the TBox $\mathcal{ALC}$ TBox $\mathcal{T}_C$ by taking concept names **Blue**, **Green**, and **Red** and taking the inclusions:

- $\top \sqsubseteq$ **Blue** $\sqcup$ **Green** $\sqcup$ **Red**
- **Blue** $\sqcap \exists r.$**Blue** $\sqsubseteq$ **Clash**
- **Red** $\sqcap \exists r.$**Red** $\sqsubseteq$ **Clash**
- **Green** $\sqcap \exists r.$**Green** $\sqsubseteq$ **Clash**

Let $F = \exists x$ **Clash**(x). Then $(\mathcal{T}_C, \mathcal{A}_G) \models F$ if, and only if, G is not 3-colorable.

Restricting the Description Logic and the Query Language

- FOPL is too expressive as a query language for knowledge bases. The combined complexity of querying even DL-Lite or $\mathcal{EL}$ knowledge bases with FOPL queries is undecidable.
- For $\mathcal{ALC}$ knowledge bases and basic Boolean queries of the form $\exists x A(x)$, (A a concept name) query answering is still non-tractable. The best algorithms for query answering in this case are extensions of the $\mathcal{ALC}$ tableaux algorithms discussed above.
- We consider
  - knowledge bases in $\mathcal{EL}$, restricted Schema.org, and DL-Lite only;
  - queries in $\mathcal{EL}$ and conjunctive queries only.

Answering $\mathcal{EL}$-Queries in $\mathcal{EL}$ Knowledge Bases

$\mathcal{EL}$ Concept Queries

An $\mathcal{EL}$ **concept query** is a Boolean query of the form

$$C(a)$$

where C is an $\mathcal{EL}$-concept and a an individual name. We develop a method for answering $\mathcal{EL}$ concept queries in knowledge bases

$$(\mathcal{T}, \mathcal{A}),$$

where $\mathcal{T}$ is a $\mathcal{EL}$-TBox and $\mathcal{A}$ a simple ABox.

Note: Then we also have a method for computing

$$\textbf{certanswer}(C(x), (\mathcal{T}, \mathcal{A})) = \{a \mid (\mathcal{T}, \mathcal{A}) \models C(a)\}$$

Fundamental Idea: reduce knowledge base querying to relational database querying

To answer the question whether

$$(\mathcal{T}, \mathcal{A}) \models C(a)$$

we construct from $(\mathcal{T}, \mathcal{A})$ a finite interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that

$$(\mathcal{T}, \mathcal{A}) \models C(a) \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{T},\mathcal{A}} \models C(a).$$

Thus, we reduce ontology based reasoning to database querying. After this construction database technology can be used to process queries.

Note: Such a reduction works only for a very limited number of ontology and query languages!

From $(\mathcal{T}, \mathcal{A})$ to $\mathcal{I}_{\mathcal{T},\mathcal{A}}$

The algorithm constructing $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a rather simple extension of the algorithm deciding concept subsumption $A \sqsubseteq_{\mathcal{T}} B$ for $\mathcal{EL}$.

Firstly, we assume again that $\mathcal{T}$ is in normal form: it consists of inclusions of the form

- $A \sqsubseteq B$, where A and B are concept names;
- $A_1 \sqcap A_2 \sqsubseteq B$, where A_1, A_2, B are concept names;
- $A \sqsubseteq \exists r.B$, where A, B are concept names;
- $\exists r.A \sqsubseteq B$, where A, B are concept names.

General Description

The domain $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ consists of

- all individual names a that occur in $\mathcal{A}$;
- objects d_A, for every concept name A in $\mathcal{T}$. (In the description of the subsumption algorithm d_A is denoted by A!)

It remains to compute

- $r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, for all role names r;
- $A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, for all concept names A.

This is done by computing functions S and R that are very similar to the functions introduced in the subsumption algorithm.

Algorithm Computing $\mathcal{I}_{\mathcal{T},\mathcal{A}}$

Given $\mathcal{T}$ in normal form and ABox $\mathcal{A}$, we compute functions S and R:

- S maps every $d \in \Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ to a set $S(d)$ of concept names.

  We then set $d \in A^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ if $A \in S(d)$;

- R maps every role name r to a set $R(r)$ of pairs (d_1, d_2) in $\Delta^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$.

  We then set $(d_1, d_2) \in r^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$ if $(d_1, d_2) \in R(r)$.

We initialise S and R as follows:

- $S(a) = \{B \mid B(a) \in \mathcal{A}\}$;
- $S(d_A) = \{A\}$ (as in the subumption algorithm, where we had $d_A = A$!)
- $R(r) = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$.

Algorithm

Apply the following four rules to $\boldsymbol{S}$ and $\boldsymbol{R}$ exhaustively:

(simpleR) If $A \in S(d)$ and $A \sqsubseteq B \in \mathcal{T}$ and $B \notin S(d)$, then

$$S(d) := S(d) \cup \{B\}.$$

(conjR) If $A_1, A_2 \in S(d)$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $B \notin S(d)$, then

$$S(d) := S(d) \cup \{B\}.$$

(rightR) If $A \in S(d)$ and $A \sqsubseteq \exists r.B \in \mathcal{T}$ and $(d, d_B) \notin R(r)$, then

$$R(r) := R(r) \cup \{(d, d_B)\}.$$

(leftR) If $(d_1, d_2) \in R(r)$ and $B \in S(d_2)$ and $\exists r.B \sqsubseteq A \in \mathcal{T}$ and $A \notin S(d_1)$, then

$$S(d_1) := S(d_1) \cup \{A\}.$$

Example

Let $\mathcal{T}$ be defined as:

$$\begin{array}{rcl}
\textbf{BasketballClub} & \sqsubseteq & \textbf{Club} \\
\textbf{BasketballPlayer} & \sqsubseteq & \exists \textbf{plays_for}.\textbf{BasketballClub} \\
\exists \textbf{plays_for}.\textbf{Club} & \sqsubseteq & \textbf{Player} \\
\textbf{Player} & \sqsubseteq & \textbf{Human_being}
\end{array}$$

Let $\mathcal{A}$ be defined as:

$$\textbf{Basketballplayer}(\textbf{bob}), \quad \textbf{Player}(\textbf{jim})$$

$$\textbf{Basketballclub}(\textbf{tigers}), \quad \textbf{Club}(\textbf{lions})$$

$$\textbf{plays\_for}(\textbf{rob}, \textbf{tigers}), \quad \textbf{plays\_for}(\textbf{bob}, \textbf{lions})$$

Construction of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$

The initial assignment (with obvious abbreviations) is given by

$$
\begin{aligned}
S(d_{\textsf{Basketclub}}) &= \{\textsf{Basketclub}\} \\
S(d_{\textsf{Basketplayer}}) &= \{\textsf{Basketplayer}\} \\
S(d_{\textsf{Club}}) &= \{\textsf{Club}\} \\
S(d_{\textsf{Player}}) &= \{\textsf{Player}\} \\
S(d_{\textsf{Human}}) &= \{\textsf{Human}\} \\
R(\textsf{plays\_for}) &= \{(\textsf{rob}, \textsf{tigers}), (\textsf{bob}, \textsf{lion})\} \\
S(\textsf{bob}) &= \{\textsf{Baskplayer}\} \\
S(\textsf{jim}) &= \{\textsf{Player}\} \\
S(\textsf{tigers}) &= \{\textsf{Baskclub}\} \\
S(\textsf{lions}) &= \{\textsf{Club}\} \\
S(\textsf{rob}) &= \emptyset
\end{aligned}
$$

Rule Applications

Now applications of (simpleR), (rightR), (leftR) are step-by-step as follows:

- Update S using (simpleR):

$$S(d_{\textbf{BaskClub}}) = \{\textbf{BaskClub}, \textbf{Club}\}.$$

- Update R using (rightR):

$$R(\textbf{plays\_for}) = \{(d_{\textbf{Baskplayer}}, d_{\textbf{BaskClub}})\}.$$

- Update S using (simpleR):

$$S(d_{\textbf{Player}}) = \{\textbf{Player}, \textbf{Human}\}.$$

- Update S using (leftR):

$$S(d_{\textbf{Baskplayer}}) = \{\textbf{Baskplayer}, \textbf{Player}\}.$$

- Update S using (simpleR):

$$S(d_{\textbf{Baskplayer}}) = \{\textbf{Baskplayer}, \textbf{Player}, \textbf{Human}\}.$$

Rule applications continued

- Update S using (simpleR):

$$S(\textbf{tigers}) = \{\textbf{BaskClub}, \textbf{Club}\}.$$

- Update S using (simpleR):

$$S(\textbf{jim}) = \{\textbf{Player}, \textbf{Human}\}.$$

- Update R using (rightR):

$$R(\textbf{plays\_for}) = \{(d_{\textbf{Baskplayer}}, d_{\textbf{BaskClub}}), (\textbf{bob}, d_{\textbf{BaskClub}})\}.$$

- Since $S(\textbf{bob})$ contains **Baskplayer**, we obtain using rules:

$$S(\textbf{bob}) = \{\textbf{Baskplayer}, \textbf{Player}, \textbf{Human}\}.$$

- Update S using (leftR):

$$S(\textbf{rob}) = \{\textbf{Player}\}.$$

- Update S using (leftR):

$$S(\textbf{rob}) = \{\textbf{Player}, \textbf{Human}\}.$$

The final assignment

$$
\begin{aligned}
S(d_{\textsf{Baskclub}}) &= \{\textsf{Baskclub}, \textsf{Club}\} \\
S(d_{\textsf{Baskplayer}}) &= \{\textsf{Baskplayer}, \textsf{Player}, \textsf{Human}\} \\
S(d_{\textsf{Club}}) &= \{\textsf{Club}\} \\
S(d_{\textsf{Player}}) &= \{\textsf{Player}, \textsf{Human}\} \\
S(d_{\textsf{Human}}) &= \{\textsf{Human}\} \\
R(\textsf{plays\_for}) &= \{(d_{\textsf{Baskplayer}}, d_{\textsf{BaskClub}}), (\textsf{rob}, \textsf{tigers}), (\textsf{bob}, \textsf{lion}), (\textsf{bob}, d_{\textsf{BaskClub}})\} \\
S(\textsf{bob}) &= \{\textsf{Baskplayer}, \textsf{Player}, \textsf{Human}\} \\
S(\textsf{jim}) &= \{\textsf{Player}\} \\
S(\textsf{tigers}) &= \{\textsf{Baskclub}\} \\
S(\textsf{lions}) &= \{\textsf{Club}\} \\
S(\textsf{rob}) &= \{\textsf{Player}, \textsf{Human}\}
\end{aligned}
$$

The interpretation $\mathcal{I}_{\mathcal{T},\mathcal{A}}$

- $\Delta^{\mathcal{I}}_{\mathcal{T},\mathcal{A}} = \{d_{\textsf{Baskclub}}, d_{\textsf{Baskplayer}}, d_{\textsf{Club}}, d_{\textsf{Player}}, d_{\textsf{Human}}, \textsf{bob}, \textsf{jim}, \textsf{tigers}, \textsf{lions}, \textsf{rob}\}$;
- $\textsf{Baskclub}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} = \{d_{\textsf{Baskclub}}, \textsf{tigers}\}$;
- $\textsf{Club}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} = \{d_{\textsf{Club}}, d_{\textsf{Baskclub}}, \textsf{tigers}\}$;
- $\textsf{Baskplayer}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} = \{d_{\textsf{Baskplayer}}, \textsf{bob}\}$;
- $\textsf{Player}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} = \{d_{\textsf{Player}}, d_{\textsf{Baskplayer}}, \textsf{bob}, \textsf{jim}, \textsf{rob}\}$;
- $\textsf{Human}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} = \{d_{\textsf{Human}}, d_{\textsf{Player}}, d_{\textsf{Baskplayer}}, \textsf{bob}, \textsf{jim}, \textsf{rob}\}$;
- $\textsf{plays\_for}^{\mathcal{I}_{\mathcal{T},\mathcal{A}}} = \{(d_{\textsf{Baskplayer}}, d_{\textsf{BaskClub}}), (\textsf{rob}, \textsf{tigers}), (\textsf{bob}, \textsf{lion}), (\textsf{bob}, d_{\textsf{BaskClub}})\}$.

Now

$$(\mathcal{T}, \mathcal{A}) \models C(a) \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{T},\mathcal{A}} \models C(a)$$

for all $\mathcal{EL}$ concepts C and a in $\mathcal{A}$. For example,

$$\mathcal{I}_{\mathcal{T},\mathcal{A}} \models \exists \textsf{plays\_for}.\textsf{Baskclub}(\textsf{bob}), \quad \mathcal{I}_{\mathcal{T},\mathcal{A}} \models \textsf{Human}(\textsf{rob})$$

Another Example

We consider the knowledge base $\mathcal{S} = (\mathcal{O}, \mathcal{B})$ given by the ABox $\mathcal{B}$ consisting of

$$\mathbf{Person}(\mathbf{john}), \quad \mathbf{Person}(\mathbf{nick}), \quad \mathbf{Person}(\mathbf{toni})$$

$$\mathbf{hasFather}(\mathbf{john}, \mathbf{nick}), \quad \mathbf{hasFather}(\mathbf{nick}, \mathbf{toni})$$

and the TBox $\mathcal{O}$ given by

$$\mathcal{O} = \{\mathbf{Person} \sqsubseteq \exists \mathbf{has\_Father}.\mathbf{Person}\}.$$

We construct $\mathcal{I}_{\mathcal{S}}$.

Constructing $\mathcal{I}_\mathcal{S}$

The initial assignment is given by

$$\begin{aligned}
S(d_{\textbf{Person}}) &= \{\textbf{Person}\}\\
S(\textbf{john}) &= \{\textbf{Person}\}\\
S(\textbf{nick}) &= \{\textbf{Person}\}\\
S(\textbf{toni}) &= \{\textbf{Person}\}\\
R(\textbf{hasFather}) &= \{(\textbf{john}, \textbf{nick}), (\textbf{nick}, \textbf{toni})\}
\end{aligned}$$

Four applications of the rule (rightR) add

$$\{(\textbf{john}, d_{\textbf{Person}}), (\textbf{nick}, d_{\textbf{Person}}), (\textbf{toni}, d_{\textbf{Person}}), (d_{\textbf{Person}}, d_{\textbf{Person}})\}$$

to the original $R(\textbf{hasFather})$. After that, no rule is applicable.

The interpretation $\mathcal{I}_\mathcal{S}$

We obtain the interpretation $\mathcal{I}_\mathcal{S}$ defined as

$$\begin{aligned}
\Delta^{\mathcal{I}_\mathcal{S}} &= \{d_{\mathsf{Person}}, \mathsf{john}, \mathsf{nick}, \mathsf{toni}\} \\
\mathsf{Person}^{\mathcal{I}_\mathcal{S}} &= \{d_{\mathsf{Person}}, \mathsf{john}, \mathsf{nick}, \mathsf{toni}\} \\
\mathsf{hasFather}^{\mathcal{I}_\mathcal{S}} &= \{(\mathsf{john}, \mathsf{nick}), (\mathsf{nick}, \mathsf{toni}), (\mathsf{john}, d_{\mathsf{Person}}), \\
&\quad (\mathsf{nick}, d_{\mathsf{Person}}), (\mathsf{toni}, d_{\mathsf{Person}}), (d_{\mathsf{Person}}, d_{\mathsf{Person}})\}
\end{aligned}$$

We have

$$\mathcal{S} \models C(a) \quad \Leftrightarrow \quad \mathcal{I}_\mathcal{S} \models C(a)$$

for all $\mathcal{EL}$ concepts C and a from $\mathcal{B}$. For example

$$\mathcal{I}_\mathcal{S} \models \exists \mathsf{hasFather}.\exists \mathsf{hasFather}.\mathsf{Person}(\mathsf{toni})$$

Answering Conjunctive Queries by Rewriting in DL-Lite

Conjunctive Queries

A FOPL query $F(x_1, \ldots, x_k)$ is a **conjunctive query** if it is constructed from atomic formulas $P(y_1, \ldots, y_n)$ using $\wedge$ and $\exists$ only.

In SQL, conjunctive queries correspond to

"Select-from-where queries",

where the "where-conditions" use only conjunctions of "$=$-conditions".

Examples

The queries

- $F(x) = \textbf{Person}(x)$;
- $F(x) = \exists y.\textbf{hasFather}(x, y)$;
- $F(x) = \exists y_1 \exists y_2 \exists y_3.(\textbf{hasFather}(x, y_1) \wedge \textbf{hasFather}(y_1, y_2); \wedge \textbf{hasFather}(y_2, y_3))$,
- $F(x, y_3) = \exists y_1 \exists y_2.(\textbf{hasFather}(x, y_1) \wedge \textbf{hasFather}(y_1, y_2) \wedge \textbf{hasFather}(y_2, y_3))$.

are conjunctive queries.

Query Rewriting for DL-Lite

Given a DL-Lite TBox $\mathcal{T}$ and a conjunctive query $F(x_1, \ldots, x_n)$ one can compute a FOPL query

$$F_{\mathcal{T}}(x_1, \ldots, x_n)$$

such that for every simple ABox $\mathcal{A}$, the database instance $\mathcal{I}_{\mathcal{A}}$ corresponding to $\mathcal{A}$, and any $a_1, \ldots, a_n$ in $\mathsf{Ind}(\mathcal{A})$ the following holds:

$$(\mathcal{T}, \mathcal{A}) \models F(a_1, \ldots, a_n) \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{A}} \models F_{\mathcal{T}}(a_1, \ldots, a_n).$$

Checking $\mathcal{I}_{\mathcal{A}} \models F_{\mathcal{T}}(a_1, \ldots, a_n)$ is again a standard database evaluation problem.

We first illustrate the construction of $F_{\mathcal{T}}(x_1, \ldots, x_n)$ using an example.

Example: Rewriting

For the TBox

$$\mathcal{T} = \{\textbf{Basketballplayer} \sqsubseteq \textbf{Player}, \textbf{Footballplayer} \sqsubseteq \textbf{Player}, \textbf{Handballplayer} \sqsubseteq \textbf{Player}\}$$

and the query

$$F(x) = \textbf{Player}(x)$$

one can take

$$F_{\mathcal{T}}(x) = \textbf{Basketballplayer}(x) \vee \textbf{Footballplayer}(x) \vee \textbf{Handballplayer}(x) \vee \textbf{Player}(x)$$

Rewriting Algorithm for Fragment DL-Lite$_{\text{tiny}}$

We give the rewriting algorithm for a small fragment DL-Lite$_{\text{tiny}}$ of DL-Lite (and Schema.org) consisting of inclusions of the form

- $A \sqsubseteq B$, where A and B are concept names;
- domain restrictions $\exists r.\top \sqsubseteq A$, where r is a role name and A a concept name;
- range restrictions $\exists r^{-}.\top \sqsubseteq A$, where r is a role name and A a concept name.

Rewriting Algorithm for Fragment DL-Lite$_{\text{tiny}}$

The rewriting algorithm computes for any

- query of the form $F(x) = A(x)$ with A a concept name and
- DL-Lite$_{\text{tiny}}$ TBox $\mathcal{T}$

a FOPL query $F_{\mathcal{T}}(x)$ such that for every simple ABox $\mathcal{A}$ and $a \in \mathbf{Ind}(\mathcal{A})$:

$$(\mathcal{T}, \mathcal{A}) \models A(a) \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{A}} \models F_{\mathcal{T}}(a)$$

The Algorithm

Assume $\mathcal{T}$ and $F(x) = A(x)$ are given. We compute sets $I(A)$, $I_R(A)$, and $I_{R^-}(A)$ which together provide 'all possible reasons for $A(a)$':

- Compute $I(A) = \{B \mid \mathcal{T} \models B \sqsubseteq A\}$ as follows: Initialise $I(A) = \{A\}$. Now apply exhaustively the following rule: if $B' \in I(A)$ and $B \sqsubseteq B' \in \mathcal{T}$ and $B \notin I(A)$, then update

$$I(A) := I(A) \cup \{B\}$$

- We obtain $I_R(A) = \{\exists r.\top \mid \mathcal{T} \models \exists r.\top \sqsubseteq A\}$ as

$$I_R(A) = \{\exists r.\top \mid \exists r.\top \sqsubseteq B \in \mathcal{T}, B \in I(A)\}$$

- We obtain $I_{R^-}(A) = \{\exists r^-.\top \mid \mathcal{T} \models \exists r^-.\top \sqsubseteq A\}$ as

$$I_{R^-}(A) = \{\exists r^-.\top \mid \exists r^-.\top \sqsubseteq B \in \mathcal{T}, B \in I(A)\}$$

The Algorithm

Then set $I(A) = \{$Person, Student$\}$

$$F_{\mathcal{T}}(x) = \bigvee_{B \in I(A)} B(x) \vee \bigvee_{\exists r.\top \in I_R(A)} \exists y r(x,y) \vee \bigvee_{\exists r.\top \in I_{R^-}(A)} \exists y r(y,x)$$

Consider $\mathcal{T}$ defined as

$$\exists \textbf{student\_at}.\top \sqsubseteq \textbf{Student}, \quad \exists \textbf{student\_at}^{-}.\top \sqsubseteq \textbf{University}$$

$$\textbf{Student} \sqsubseteq \textbf{Person}, \quad \textbf{University} \sqsubseteq \textbf{Institution}$$

For $F(x) = \textbf{Person}(x)$ we obtain

$$F_{\mathcal{T}}(x) = \textbf{Person}(x) \vee \textbf{Student}(x) \vee \exists y \textbf{student\_at}(x,y)$$