QUERY ANSWERING WITH DESCRIPTION LOGIC ONTOLOGIES

Meghyn Bienvenu (*CNRS & Université de Montpellier*)
Magdalena Ortiz (*Vienna University of Technology*)

INSTANCE QUERIES

Instance queries (IQs): find instances of a given concept or role

$\mathsf{A}(\mathsf{x})$ where $A \in \mathsf{N_C}$ — **concept** instance query

$\mathsf{r}(\mathsf{x}, \mathsf{y})$ where $r \in \mathsf{N_R}$ — **role** instance query

Instance queries (IQs): find instances of a given concept or role

$\mathsf{A}(\mathsf{x})$ where $A \in \mathsf{N_C}$	**concept** instance query
$\mathsf{r}(\mathsf{x},\mathsf{y})$ where $r \in \mathsf{N_R}$	**role** instance query

To query for a **complex concept** C, take $A_C(x)$ for fresh $A_C \in \mathsf{N_C}$ and add $C \sqsubseteq A_C$ to the TBox

Instance queries (IQs): find instances of a given concept or role

$\mathsf{A}(\mathsf{x})$ where $A \in \mathsf{N_C}$	concept instance query
$\mathsf{r}(\mathsf{x},\mathsf{y})$ where $r \in \mathsf{N_R}$	role instance query

To query for a complex concept C, take $A_C(x)$ for fresh $A_C \in \mathsf{N_C}$ and add $C \sqsubseteq A_C$ to the TBox

Remarks:

- Instance query answering is often called *instance checking*
- Focus of OMQA until mid-2000s

Input = instance query q + DL-Lite$_R$ TBox $\mathcal{T}$

We construct an FO-rewriting of **q** w.r.t. $\mathcal{T}$

More specifically, we construct:

- an FO-rewriting of q relative to consistent ABoxes, and
- an FO-rewriting of unsatisfiability

(these can be easily combined into FO-rewriting of q for *all* ABoxes)

We first define two procedures:

ComputeSubsumees *all reasons for an individual to be in B*

input concept B, TBox $\mathcal{T}$

output set of C such that $\mathcal{T} \models C \sqsubseteq B$ $\rightsquigarrow$ **subsumees** of B w.r.t. $\mathcal{T}$

ComputeSubroles *all reasons for a pair to be in R*

input role R, TBox $\mathcal{T}$

output set of S such that $\mathcal{T} \models S \sqsubseteq R$ $\rightsquigarrow$ **subroles** of R w.r.t. $\mathcal{T}$

Algorithm ComputeSubsumees

INPUT: DL-Lite$_R$ TBox $\mathcal{T}$, concept $B \in \mathsf{N_C} \cup \{\exists R \mid R \in \mathsf{N_R^{\pm}}\}$

1. **Initialize** Subsumees $= \{B\}$ and Examined $= \emptyset$.
2. While Subsumees $\setminus$ Examined $\neq \emptyset$
   - 2.1 **Select** $D \in$ Subsumees $\setminus$ Examined and add D to Examined.
   - 2.2 For every concept inclusion $C \sqsubseteq D \in \mathcal{T}$
     - If $C \notin$ Subsumees, **add** C **to** Subsumees
   - 2.3 For every role inclusion $R \sqsubseteq S \in \mathcal{T}$ such that $D = \exists S$.
     - If $\exists R \notin$ Subsumees, **add** $\exists R$ **to** Subsumees
   - 2.4 For every role inclusion $R \sqsubseteq S \in \mathcal{T}$ such that $D = \exists \mathrm{inv}(S)$.
     - If $\exists \mathrm{inv}(R) \notin$ Subsumees, **add** $\exists \mathrm{inv}(R)$ **to** Subsumees.
3. **Return** Subsumees.

COMPUTING SUBSUMEES: AN EXAMPLE (1/3)

ComputeSubsumees on $(\mathcal{T}, \mathsf{Dish})$, where $\mathcal{T}$:

$$\begin{aligned}
\mathsf{ItalDish} &\sqsubseteq \mathsf{Dish}\\
\mathsf{VegDish} &\sqsubseteq \mathsf{Dish}\\
\mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred}\\
\exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish}\\
\mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse}\\
\mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse}
\end{aligned}$$

COMPUTING SUBSUMEES: AN EXAMPLE (1/3)

ComputeSubsumees on $(\mathcal{T}, \mathsf{Dish})$, where $\mathcal{T}$:

$$
\begin{aligned}
\mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\
\mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\
\mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\
\exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\
\mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\
\mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Examined} &= \emptyset \\
\mathsf{Subsumees} &= \{\mathsf{Dish}\}
\end{aligned}
$$

COMPUTING SUBSUMEES: AN EXAMPLE (1/3)

ComputeSubsumees on $(\mathcal{T}, \text{Dish})$, where $\mathcal{T}$:

$$\begin{aligned} \text{ItalDish} &\sqsubseteq \text{Dish} \\ \text{VegDish} &\sqsubseteq \text{Dish} \\ \text{Dish} &\sqsubseteq \exists \text{hasIngred} \\ \exists \text{hasCourse}^{-} &\sqsubseteq \text{Dish} \\ \text{hasMain} &\sqsubseteq \text{hasCourse} \\ \text{hasDessert} &\sqsubseteq \text{hasCourse} \end{aligned}$$

$$\begin{aligned} \text{Examined} &= \emptyset \\ \text{Subsumees} &= \{\text{Dish}\} \end{aligned}$$

Choose: Dish

$$\begin{aligned} \text{Examined} &= \{\text{Dish}\} \\ \text{Subsumees} &= \{\text{Dish}, \text{ItalDish}, \text{VegDish}, \exists \text{hasCourse}^{-}\} \end{aligned}$$

COMPUTING SUBSUMEES: AN EXAMPLE (1/3)

ComputeSubsumees on $(\mathcal{T}, \text{Dish})$, where $\mathcal{T}$:

$$
\begin{aligned}
\text{ItalDish} &\sqsubseteq \text{Dish} \\
\text{VegDish} &\sqsubseteq \text{Dish} \\
\text{Dish} &\sqsubseteq \exists\text{hasIngred} \\
\exists\text{hasCourse}^- &\sqsubseteq \text{Dish} \\
\text{hasMain} &\sqsubseteq \text{hasCourse} \\
\text{hasDessert} &\sqsubseteq \text{hasCourse}
\end{aligned}
$$

	Examined $= \emptyset$ Subsumees $= \{\text{Dish}\}$
Choose: Dish	Examined $= \{\text{Dish}\}$ Subsumees $= \{\text{Dish}, \text{ItalDish}, \text{VegDish}, \exists\text{hasCourse}^-\}$
Choose: ItalDish	Examined $= \{\text{Dish}, \text{ItalDish}\}$ Subsumees $= \{\text{Dish}, \text{ItalDish}, \text{VegDish}, \exists\text{hasCourse}^-\}$

COMPUTING SUBSUMEES: AN EXAMPLE (2/3)

ComputeSubsumees on $(\mathcal{T}, \mathsf{Dish})$, where $\mathcal{T}$:

$$\begin{aligned}
\mathsf{ItalDish} &\sqsubseteq \mathsf{Dish}\\
\mathsf{VegDish} &\sqsubseteq \mathsf{Dish}\\
\mathsf{Dish} &\sqsubseteq \exists\mathsf{hasIngred}\\
\exists\mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish}\\
\mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse}\\
\mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse}
\end{aligned}$$

Choose: VegDish

$$\begin{aligned}
\text{Examined} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}\}\\
\text{Subsumees} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists\mathsf{hasCourse}^-\}
\end{aligned}$$

ComputeSubsumees on $(\mathcal{T}, \mathsf{Dish})$, where $\mathcal{T}$:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Choose: VegDish	Examined $= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}\}$
	Subsumees $= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists \mathsf{hasCourse}^-\}$

Choose: $\exists$hasCourse$^-$	Examined $= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists \mathsf{hasCourse}^-\}$
	Subsumees $= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists \mathsf{hasCourse}^-, \exists \mathsf{hasMain}^-, \exists \mathsf{hasDessert}^-\}$

COMPUTING SUBSUMEES: AN EXAMPLE (3/3)

ComputeSubsumees on $(\mathcal{T}, \mathsf{Dish})$, where $\mathcal{T}$:

$$
\begin{aligned}
\mathsf{ItalDish} &\sqsubseteq \mathsf{Dish}\\
\mathsf{VegDish} &\sqsubseteq \mathsf{Dish}\\
\mathsf{Dish} &\sqsubseteq \exists\mathsf{hasIngred}\\
\exists\mathsf{hasCourse}^{-} &\sqsubseteq \mathsf{Dish}\\
\mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse}\\
\mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse}
\end{aligned}
$$

Choose: $\exists\mathsf{hasMain}^{-}$

$$
\begin{aligned}
\text{Examined} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists\mathsf{hasCourse}^{-},\\
&\quad \exists\mathsf{hasMain}^{-}\}\\
\text{Subsumees} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists\mathsf{hasCourse}^{-},\\
&\quad \exists\mathsf{hasMain}^{-}, \exists\mathsf{hasDessert}^{-}\}
\end{aligned}
$$

COMPUTING SUBSUMEES: AN EXAMPLE (3/3)

ComputeSubsumees on $(\mathcal{T}, \mathsf{Dish})$, where $\mathcal{T}$:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Choose: $\exists \mathsf{hasMain}^-$

$$\begin{aligned} \text{Examined} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists \mathsf{hasCourse}^-, \\ &\quad \exists \mathsf{hasMain}^-\} \\ \text{Subsumees} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists \mathsf{hasCourse}^-, \\ &\quad \exists \mathsf{hasMain}^-, \exists \mathsf{hasDessert}^-\} \end{aligned}$$

Choose: $\exists \mathsf{hasDessert}^-$

$$\begin{aligned} \text{Examined} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists \mathsf{hasCourse}^-, \\ &\quad \exists \mathsf{hasMain}^-, \exists \mathsf{hasDessert}^-\} \\ \text{Subsumees} &= \{\mathsf{Dish}, \mathsf{ItalDish}, \mathsf{VegDish}, \exists \mathsf{hasCourse}^-, \\ &\quad \exists \mathsf{hasMain}^-, \exists \mathsf{hasDessert}^-\} \end{aligned}$$

Algorithm ComputeSubroles

Input: DL-Lite$_R$ TBox $\mathcal{T}$, role $R \in \mathsf{N}_\mathsf{R}^\pm$

1. **Initialize** Subroles $= \{R\}$ and Examined $= \emptyset$.
2. While Subroles $\setminus$ Examined $\neq \emptyset$
   2.1 **Select** $S \in$ Subroles $\setminus$ Examined and add S to Examined.
   2.2 For every role inclusion $U \sqsubseteq S$ **or** $\mathrm{inv}(U) \sqsubseteq \mathrm{inv}(S)$ in $\mathcal{T}$
      - If $U \notin$ Subsumees, **add** U **to** Subsumees
3. **Return** Subroles.

Algorithm ComputeSubroles
Input: DL-Lite$_R$ TBox $\mathcal{T}$, role $R \in \mathsf{N}_\mathsf{R}^\pm$

1. **Initialize** Subroles $= \{R\}$ and Examined $= \emptyset$.
2. While Subroles $\setminus$ Examined $\neq \emptyset$
   2.1 **Select** $S \in$ Subroles $\setminus$ Examined and add S to Examined.
   2.2 For every role inclusion $U \sqsubseteq S$ **or** $\mathsf{inv}(U) \sqsubseteq \mathsf{inv}(S)$ in $\mathcal{T}$
      - If $U \notin$ Subsumees, **add** U **to** Subsumees
3. **Return** Subroles.

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

Run on hasCourse:

$$\text{Subroles} = \{\mathsf{hasCourse}, \mathsf{hasMain}, \mathsf{hasDessert}\}$$

For a concept C and variable x, **define** $C(x)$ as follows:

- if $C = A \in \mathsf{N_C}$, then $C(x) = A(x)$
- if $C = \exists r$, then $C(x) = \exists z\, r(x, z)$
- if $C = \exists r^-$, then $C(x) = \exists z\, r(z, x)$

For a role R and variables x, y, **define** $R(x, y)$ as follows:

- if $R = r \in \mathsf{N_R}$, then $R(x, y) = r(x, y)$
- if $R = r^-$, then $R(x, y) = r(y, x)$

FROM CONCEPTS AND ROLES TO QUERIES (2)

Let SC $=$ ComputeSubsumees$(A, \mathcal{T})$, SR $=$ ComputeSubroles$(r, \mathcal{T})$.

Rewriting of $A(x)$ w.r.t. $\mathcal{T}$ (and consistent ABoxes):

$$\text{RewriteIQ}(A, \mathcal{T}) = \bigvee_{\mathsf{C} \in \mathsf{SC}} \mathsf{C}(\mathsf{x})$$

Let **SC** $=$ ComputeSubsumees$(A, \mathcal{T})$, **SR** $=$ ComputeSubroles$(r, \mathcal{T})$.

Rewriting of $A(x)$ w.r.t. $\mathcal{T}$ (and consistent ABoxes):

$$\text{RewriteIQ}(A, \mathcal{T}) = \bigvee_{\mathsf{C} \in \mathbf{SC}} \mathsf{C}(\mathsf{x})$$

Rewriting of $r(x, y)$ w.r.t. $\mathcal{T}$ (and consistent ABoxes):

$$\text{RewriteIQ}(r, \mathcal{T}) = \bigvee_{R \in \mathbf{SR}} R(x, y)$$

Let $\mathsf{SC} = \text{ComputeSubsumees}(A, \mathcal{T})$, $\mathsf{SR} = \text{ComputeSubroles}(r, \mathcal{T})$.

Rewriting of $A(x)$ w.r.t. $\mathcal{T}$ (and consistent ABoxes):

$$\text{RewriteIQ}(A, \mathcal{T}) = \bigvee_{\mathsf{C} \in \mathsf{SC}} \mathsf{C}(\mathsf{x})$$

Rewriting of $r(x, y)$ w.r.t. $\mathcal{T}$ (and consistent ABoxes):

$$\text{RewriteIQ}(r, \mathcal{T}) = \bigvee_{R \in \mathsf{SR}} R(x, y)$$

The rewriting is ABox-independent and polysize in $|\mathcal{T}|$ and $|q|$.

We have already computed:

$$\text{ComputeSubsumees}(\text{Dish}, \mathcal{T}) = \{\text{Dish}, \text{ItalDish}, \text{VegDish}, \exists\text{hasCourse}^{-}, \exists\text{hasMain}^{-}, \exists\text{hasDessert}^{-}\}$$

Get following **rewriting of** Dish(x) **w.r.t.** $\mathcal{T}$ (for consistent ABoxes):

$$\begin{aligned}\text{RewriteIQ}(\text{Dish}, \mathcal{T}) = \text{Dish}(x) &\vee \text{ItalDish}(x) \vee \text{VegDish}(x)\\ &\vee \exists y.\text{hasCourse}(y, x) \vee \exists y.\text{hasMain}(y, x)\\ &\vee \exists y.\text{hasDessert}(y, x)\end{aligned}$$

EXAMPLE OF QUERY REWRITING (2/2)

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

ABox $\mathcal{A}$:

$\mathsf{hasMain}(m, d_1)$
$\mathsf{hasDessert}(m, d_2)$
$\mathsf{VegDish}(d_3)$

$$\begin{aligned} \mathsf{RewriteIQ}(\mathsf{Dish}, \mathcal{T}) = {} & \mathsf{Dish}(x) \lor \mathsf{ItalDish}(x) \lor \mathsf{VegDish}(x) \lor \exists y.\mathsf{hasCourse}(y, x) \\ & \lor \exists y.\mathsf{hasMain}(y, x) \lor \exists y.\mathsf{hasDessert}(y, x) \end{aligned}$$

Certain answers:

$$\begin{aligned} \mathsf{ItalDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{VegDish} &\sqsubseteq \mathsf{Dish} \\ \mathsf{Dish} &\sqsubseteq \exists \mathsf{hasIngred} \\ \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \end{aligned}$$

ABox $\mathcal{A}$:

hasMain(m, d_1)
hasDessert(m, d_2)
VegDish(d_3)

$$\mathsf{RewriteIQ}(\mathsf{Dish}, \mathcal{T}) = \mathsf{Dish}(x) \lor \mathsf{ItalDish}(x) \lor \mathsf{VegDish}(x) \lor \exists y.\mathsf{hasCourse}(y, x) \lor \exists y.\mathsf{hasMain}(y, x) \lor \exists y.\mathsf{hasDessert}(y, x)$$

Certain answers: d_1, because of the disjunct $\exists y.\mathsf{hasMain}(y, x)$
d_2, because of the disjunct $\exists y.\mathsf{hasDessert}(y, x)$
d_3, because of the disjunct $\mathsf{VegDish}(x)$

CHECKING UNSATISFIABILITY

We have an FO-rewriting of q w.r.t. $\mathcal{T}$ relative to consistent ABoxes

To obtain a rewriting of q that works for all ABoxes,
we need a rewriting of unsatisfiability

We have an FO-rewriting of q w.r.t. $\mathcal{T}$ relative to consistent ABoxes

To obtain a rewriting of q that works for all ABoxes,
we need a rewriting of unsatisfiability

Main ideas:

- only negative inclusions are relevant for detecting contradictions
- create one subquery for each negation inclusion $G \sqsubseteq \neg H$
- consider all possible ways of violating $G \sqsubseteq \neg H$: combinations of a subsumee (subrole) of G and a subsumee (subrole) of H

For a negative concept inclusion $A \sqsubseteq \neg B$:

$$\mathrm{RewriteNeg}(A, B, \mathcal{T}) = \bigvee_{\substack{C \in \mathrm{ComputeSubsumees}(A,\mathcal{T}) \\ D \in \mathrm{ComputeSubsumees}(B,\mathcal{T})}} \exists \mathsf{x}.(\mathsf{C}(\mathsf{x}) \wedge \mathsf{D}(\mathsf{x}))$$

For a negative concept inclusion $A \sqsubseteq \neg B$:

$$\mathrm{RewriteNeg}(A, B, \mathcal{T}) = \bigvee_{\substack{C \in \mathrm{ComputeSubsumees}(A, \mathcal{T}) \\ D \in \mathrm{ComputeSubsumees}(B, \mathcal{T})}} \exists \mathsf{x}.(\mathsf{C}(\mathsf{x}) \land \mathsf{D}(\mathsf{x}))$$

For a negative role inclusion $R \sqsubseteq \neg S$:

$$\mathrm{RewriteNeg}(R, S, \mathcal{T}) = \bigvee_{\substack{U \in \mathrm{ComputeSubroles}(R, \mathcal{T}) \\ V \in \mathrm{ComputeSubroles}(S, \mathcal{T})}} \exists \mathsf{x}, \mathsf{y}.(\mathsf{U}(\mathsf{x}, \mathsf{y}) \land \mathsf{V}(\mathsf{x}, \mathsf{y}))$$

REWRITING OF UNSATISFIABILITY

For a negative concept inclusion $A \sqsubseteq \neg B$:

$$\mathrm{RewriteNeg}(A, B, \mathcal{T}) = \bigvee_{\substack{C \in \mathrm{ComputeSubsumees}(A, \mathcal{T}) \\ D \in \mathrm{ComputeSubsumees}(B, \mathcal{T})}} \exists \mathsf{x}.(\mathsf{C}(\mathsf{x}) \wedge \mathsf{D}(\mathsf{x}))$$

For a negative role inclusion $R \sqsubseteq \neg S$:

$$\mathrm{RewriteNeg}(R, S, \mathcal{T}) = \bigvee_{\substack{U \in \mathrm{ComputeSubroles}(R, \mathcal{T}) \\ V \in \mathrm{ComputeSubroles}(S, \mathcal{T})}} \exists \mathsf{x}, \mathsf{y}.(\mathsf{U}(\mathsf{x}, \mathsf{y}) \wedge \mathsf{V}(\mathsf{x}, \mathsf{y}))$$

For a TBox, following Boolean query checks for unsatisfiability:

$$\mathrm{RewriteUnsat}(\mathcal{T}) = \bigvee_{\mathsf{G} \sqsubseteq \neg \mathsf{H} \in \mathcal{T}} \mathrm{RewriteNeg}(\mathsf{G}, \mathsf{H}, \mathcal{T})$$

EXAMPLE: REWRITING OF UNSATISFIABILITY

Let $\mathcal{T}$ be the following TBox:

$$\begin{aligned} \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasMain} &\sqsubseteq \neg \mathsf{hasDessert} \\ \mathsf{Dish} &\sqsubseteq \neg \exists \mathsf{hasCourse} \end{aligned}$$

Let $\mathcal{T}$ be the following TBox:

$$
\begin{aligned}
\exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\
\mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\
\mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \\
\mathsf{hasMain} &\sqsubseteq \neg \mathsf{hasDessert} \\
\mathsf{Dish} &\sqsubseteq \neg \exists \mathsf{hasCourse}
\end{aligned}
$$

Queries testing violation of two negative inclusions:

$\mathsf{RewriteNeg}(\mathsf{hasMain}, \mathsf{hasDessert}, \mathcal{T}) =$

$\mathsf{RewriteNeg}(\mathsf{Dish}, \exists \mathsf{hasCourse}, \mathcal{T}) =$

EXAMPLE: REWRITING OF UNSATISFIABILITY

Let $\mathcal{T}$ be the following TBox:

$$\begin{aligned} \exists\mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasMain} &\sqsubseteq \neg\mathsf{hasDessert} \\ \mathsf{Dish} &\sqsubseteq \neg\exists\mathsf{hasCourse} \end{aligned}$$

Queries testing violation of two negative inclusions:

$$\mathsf{RewriteNeg}(\mathsf{hasMain}, \mathsf{hasDessert}, \mathcal{T}) = \exists x, y\ \mathsf{hasMain}(\mathbf{x}, \mathbf{y}) \wedge \mathsf{hasDessert}(\mathbf{x}, \mathbf{y})$$

$$\mathsf{RewriteNeg}(\mathsf{Dish}, \exists\mathsf{hasCourse}, \mathcal{T}) =$$

Let $\mathcal{T}$ be the following TBox:

$$\begin{aligned} \exists \mathsf{hasCourse}^- &\sqsubseteq \mathsf{Dish} \\ \mathsf{hasMain} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasDessert} &\sqsubseteq \mathsf{hasCourse} \\ \mathsf{hasMain} &\sqsubseteq \neg \mathsf{hasDessert} \\ \mathsf{Dish} &\sqsubseteq \neg \exists \mathsf{hasCourse} \end{aligned}$$

Queries testing violation of two negative inclusions:

$$\mathsf{RewriteNeg}(\mathsf{hasMain}, \mathsf{hasDessert}, \mathcal{T}) = \exists x, y\ \mathsf{hasMain}(\mathsf{x}, \mathsf{y}) \wedge \mathsf{hasDessert}(\mathsf{x}, \mathsf{y})$$

$$\begin{aligned} \mathsf{RewriteNeg}(\mathsf{Dish}, \exists \mathsf{hasCourse}, \mathcal{T}) = \exists x & \bigvee_{r \in \{\mathsf{hC}, \mathsf{hM}, \mathsf{hD}\}} (\mathsf{Dish}(\mathsf{x}) \wedge \exists \mathsf{y}\ \mathsf{r}(\mathsf{x}, \mathsf{y})) \\ & \vee \bigvee_{r_1, r_2 \in \{\mathsf{hC}, \mathsf{hM}, \mathsf{hD}\}} (\exists \mathsf{y}\ \mathsf{r}_1(\mathsf{y}, \mathsf{x}) \wedge \exists \mathsf{z}\ \mathsf{r}_2(\mathsf{x}, \mathsf{z})) \end{aligned}$$

Recall: **KB unsat** $\Rightarrow$ return all tuples of ABox individuals as answers

Recall: **KB unsat** $\Rightarrow$ return all tuples of ABox individuals as answers

Σ: finite set of concept / role names that can be used in the ABox

Define unary query that **retrieves all individuals in Σ-ABox**:

$$q_{\text{ind}}^{\Sigma}(x) = \bigvee_{A \in \boldsymbol{\Sigma} \cap \mathsf{N_C}} A(x) \vee \bigvee_{r \in \boldsymbol{\Sigma} \cap \mathsf{N_R}} \exists y.(r(x,y) \vee r(y,x))$$

Recall: **KB unsat** $\Rightarrow$ return all tuples of ABox individuals as answers

Σ: finite set of concept / role names that can be used in the ABox

Define unary query that **retrieves all individuals in Σ-ABox**:

$$q^{\Sigma}_{\text{ind}}(x) = \bigvee_{A \in \boldsymbol{\Sigma} \cap \mathsf{N_C}} A(x) \vee \bigvee_{r \in \boldsymbol{\Sigma} \cap \mathsf{N_R}} \exists y.(r(x,y) \vee r(y,x))$$

Rewriting of IQ $B(x)$ w.r.t. $\mathcal{T}$ for arbitrary Σ-ABoxes:

$$\text{RewriteIQ}(B, \mathcal{T}) \vee (\text{RewriteUnsat}(\mathcal{T}) \wedge q^{\Sigma}_{\text{ind}})$$

In data complexity

- rewriting takes constant time, yields FO query
- upper bound from FO query evaluation: AC_0

In data complexity

- rewriting takes constant time, yields FO query
- upper bound from FO query evaluation: AC_0

In combined complexity:

- P membership: rewriting and evaluation both in polynomial time
- NLogSpace upper bound: ‘guess’ relevant part of rewriting

In data complexity

- rewriting takes constant time, yields FO query
- upper bound from FO query evaluation: AC_0

In combined complexity:

- P membership: rewriting and evaluation both in polynomial time
- NLogSpace upper bound: ‘guess’ relevant part of rewriting

Theorem In DL-Lite$_R$, satisfiability and instance checking are

1. in AC_0 **for data complexity**
2. **NLogSpace-complete for combined complexity.**

Note: Same bounds hold for several other DL-Lite dialects

Next consider instance checking in $\mathcal{EL}$.

Assume $\mathcal{EL}$ TBoxes given in normal form: axioms of the forms

$$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

Next consider instance checking in $\mathcal{EL}$.

Assume $\mathcal{EL}$ TBoxes given in normal form: axioms of the forms

$$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

Normalization in polytime, can introduce new concept names

$$\mathsf{A} \sqsubseteq \exists \mathsf{r}.\exists \mathsf{s}.\mathsf{D} \quad \rightsquigarrow \quad \mathsf{A} \sqsubseteq \exists \mathsf{r}.\mathsf{N}, \mathsf{N} \sqsubseteq \exists \mathsf{s}.\mathsf{D}, \exists \mathsf{s}.\mathsf{D} \sqsubseteq \mathsf{N}$$

Next consider instance checking in $\mathcal{EL}$.

Assume $\mathcal{EL}$ TBoxes given in normal form: axioms of the forms

$$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

Normalization in polytime, can introduce new concept names

$$\mathsf{A} \sqsubseteq \exists \mathsf{r}.\exists \mathsf{s}.\mathsf{D} \quad \rightsquigarrow \quad \mathsf{A} \sqsubseteq \exists \mathsf{r}.\mathsf{N}, \mathsf{N} \sqsubseteq \exists \mathsf{s}.\mathsf{D}, \exists \mathsf{s}.\mathsf{D} \sqsubseteq \mathsf{N}$$

Cannot use FO query rewriting approach for $\mathcal{EL}$:

no FO-rewriting of $A(x)$ w.r.t. $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$

Next consider instance checking in $\mathcal{EL}$.

Assume $\mathcal{EL}$ TBoxes given in normal form: axioms of the forms

$$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

Normalization in polytime, can introduce new concept names

$$\mathsf{A} \sqsubseteq \exists \mathsf{r}.\exists \mathsf{s}.\mathsf{D} \quad \rightsquigarrow \quad \mathsf{A} \sqsubseteq \exists \mathsf{r}.\mathsf{N}, \mathsf{N} \sqsubseteq \exists \mathsf{s}.\mathsf{D}, \exists \mathsf{s}.\mathsf{D} \sqsubseteq \mathsf{N}$$

Cannot use FO query rewriting approach for $\mathcal{EL}$:

no FO-rewriting of $A(x)$ w.r.t. $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$

We present a saturation-based approach.

TBox rules

$$\frac{A \sqsubseteq B_i \ (1 \leq i \leq n) \quad B_1 \sqcap \ldots \sqcap B_n \sqsubseteq D}{A \sqsubseteq D}\ \text{T1} \qquad \frac{A \sqsubseteq B \quad B \sqsubseteq \exists r.D}{A \sqsubseteq \exists r.D}\ \text{T2}$$

$$\frac{A \sqsubseteq \exists r.B \quad B \sqsubseteq D \quad \exists r.D \sqsubseteq E}{A \sqsubseteq E}\ \text{T3}$$

ABox rules

$$\frac{A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A_i(a) \ (1 \leq i \leq n)}{B(a)}\ \text{A1} \qquad \frac{\exists r.B \sqsubseteq A \quad r(a,b) \quad B(b)}{A(a)}\ \text{A2}$$

Algorithm: apply rules exhaustively, check if $A(a)$ ($r(a,b)$) is present

EXAMPLE: SATURATION IN EL

$$\mathsf{PenneArrabiata} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{ArrabiataSauce} \quad (1)$$

$$\mathsf{PenneArrabiata} \sqsubseteq \mathsf{PastaDish} \quad (2)$$

$$\mathsf{PastaDish} \sqsubseteq \mathsf{Dish} \quad (3)$$

$$\mathsf{PastaDish} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{Pasta} \quad (4)$$

$$\mathsf{ArrabiataSauce} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{Peperoncino} \quad (5)$$

$$\mathsf{Peperoncino} \sqsubseteq \mathsf{Spicy} \quad (6)$$

$$\exists \mathsf{hasIngred}.\mathsf{Spicy} \sqsubseteq \mathsf{Spicy} \quad (7)$$

$$\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish} \quad (8)$$

$$\mathsf{PenneArrabiata}(p). \quad (9)$$

EXAMPLE: SATURATION IN EL

$$\mathsf{PenneArrabiata} \sqsubseteq \exists \mathsf{hasIngred.ArrabiataSauce} \quad (1)$$
$$\mathsf{PenneArrabiata} \sqsubseteq \mathsf{PastaDish} \quad (2)$$
$$\mathsf{PastaDish} \sqsubseteq \mathsf{Dish} \quad (3)$$
$$\mathsf{PastaDish} \sqsubseteq \exists \mathsf{hasIngred.Pasta} \quad (4)$$
$$\mathsf{ArrabiataSauce} \sqsubseteq \exists \mathsf{hasIngred.Peperoncino} \quad (5)$$
$$\mathsf{Peperoncino} \sqsubseteq \mathsf{Spicy} \quad (6)$$
$$\exists \mathsf{hasIngred.Spicy} \sqsubseteq \mathsf{Spicy} \quad (7)$$
$$\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish} \quad (8)$$
$$\mathsf{PenneArrabiata}(p). \quad (9)$$

$\mathsf{ArrabSauce} \sqsubseteq \mathsf{Spicy}$	**T3** : (5), (6), (7)	(10)
$\mathsf{PenneArrab} \sqsubseteq \mathsf{Spicy}$	**T3** : **(1)**, (10), **(7)**	(11)
$\mathsf{PenneArrab} \sqsubseteq \mathsf{Dish}$	**T1** : **(2)**, **(3)**	(12)
$\mathsf{PenneArrab} \sqsubseteq \exists \mathsf{hasIngred.Pasta}$	**T2** : **(2)**, **(4)**	(13)
$\mathsf{PenneArrab} \sqsubseteq \mathsf{SpicyDish}$	**T1** : (11), (12), **(8)**	(14)
$\mathsf{Spicy}(\mathbf{p})$	**A1** : (11), **(9)**	(15)
$\mathsf{Dish}(\mathbf{p})$	**A1** : (12), **(9)**	(16)
$\mathsf{SpicyDish}(\mathbf{p})$	**A1** : (16), (15)	(17)

Saturation approach is sound: everything derived is entailed

Saturation approach is sound: everything derived is entailed

Also complete for instance checking:

Theorem Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base, and let $\mathcal{K}'$ be the result of saturating $\mathcal{K}$. For every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathcal{K}'$$

Saturation approach is sound: everything derived is entailed

Also complete for instance checking:

Theorem Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base, and let $\mathcal{K}'$ be the result of saturating $\mathcal{K}$. For every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathcal{K}'$$

Note: does not make all consequences explicit

- can have infinitely many implied axioms $\rightsquigarrow$ would not terminate!
- so: only complete for some reasoning tasks

Saturation approach is sound: everything derived is entailed

Also complete for instance checking:

Theorem Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base, and let $\mathcal{K}'$ be the result of saturating $\mathcal{K}$. For every ABox assertion α, we have:

$$\mathcal{K} \models \alpha \qquad \text{iff} \qquad \alpha \in \mathcal{K}'$$

Note: does not make all consequences explicit

- can have infinitely many implied axioms $\rightsquigarrow$ would not terminate!
- so: only complete for some reasoning tasks

Runs in polynomial time in $|\mathcal{K}|$. This is optimal:

Theorem Instance checking in $\mathcal{EL}$ is P-complete for both data and combined complexity.

Reduction from P-complete **Boolean circuit evaluation** problem

- Circuit is given as an ABox
  - one **individual name** per circuit **gate**
  - concept names And and Or indicate **type of gate**
  - concept name True marks **input gates with value 1**
  - role names leftInput and rightInput are used to **link gates**
- Same TBox for all circuits to propagate values:

$$\mathsf{Or} \sqcap \exists \mathsf{leftInput}.\mathsf{True} \sqsubseteq \mathsf{True} \qquad \mathsf{Or} \sqcap \exists \mathsf{rightInput}.\mathsf{True} \sqsubseteq \mathsf{True}$$

$$\mathsf{And} \sqcap \exists \mathsf{leftInput}.\mathsf{True} \sqcap \exists \mathsf{rightInput}.\mathsf{True} \sqsubseteq \mathsf{True}$$

Can show: **circuit outputs 1** $\Leftrightarrow$ **output gate** is answer to $\mathsf{True}(x)$

Saturation approach can be **extended to $\mathcal{ELHI}_\bot$**

Additional rules required

Key difference: new **conjunctions of concepts** can occur

$$\frac{A \sqsubseteq \exists R.D \quad \exists R^-.B \sqsubseteq E}{}$$

Saturation approach can be **extended to $\mathcal{ELHI}_\bot$**

Additional rules required

Key difference: new **conjunctions of concepts** can occur

$$\frac{A \sqsubseteq \exists R.D \quad \exists R^-.B \sqsubseteq E}{A \sqcap B \sqsubseteq \exists R.(D \sqcap E)}$$

TBox rules M, N, N' conjunctions of concept names

$$\frac{\{A \sqsubseteq B_i\}_{i=1}^{n} \quad B_1 \sqcap \ldots \sqcap B_n \sqsubseteq D}{A \sqsubseteq D}\ \text{T1} \qquad \frac{R \sqsubseteq S \quad S \sqsubseteq T}{R \sqsubseteq T}\ \text{T4} \qquad \frac{M \sqsubseteq \exists R.(N \sqcap \bot)}{M \sqsubseteq \bot}\ \text{T5}$$

$$\frac{M \sqsubseteq \exists R.(N \sqcap N') \quad N \sqsubseteq A}{M \sqsubseteq \exists R.(N \sqcap N' \sqcap A)}\ \text{T6} \qquad \frac{M \sqsubseteq \exists R.(N \sqcap A) \quad \exists S.A \sqsubseteq B \quad R \sqsubseteq S}{M \sqsubseteq B}\ \text{T7}$$

$$\frac{M \sqsubseteq \exists R.N \quad \exists \mathrm{inv}(S).A \sqsubseteq B \quad R \sqsubseteq S}{M \sqcap A \sqsubseteq \exists R.(N \sqcap B)}\ \text{T8}$$

ABox rules

$$\frac{A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \quad A_i(a)\ (1 \leq i \leq n)}{B(a)}\ \text{A1} \qquad \frac{\exists r.B \sqsubseteq A \quad r(a,b) \quad B(b)}{A(a)}\ \text{A2}$$

$$\frac{\exists r^-.B \sqsubseteq A \quad r(b,a) \quad B(b)}{A(a)}\ \text{A3} \qquad \frac{r \sqsubseteq s \quad r(a,b)}{s(a,b)}\ \text{A4} \qquad \frac{r \sqsubseteq s^- \quad r(a,b)}{s(b,a)}\ \text{A5}$$

Saturation approach can be extended to $\mathcal{ELHI}_\bot$

Additional rules required

Key difference: new conjunctions of concepts can occur

$$\frac{A \sqsubseteq \exists R.D \quad \exists R^-.B \sqsubseteq E}{A \sqcap B \sqsubseteq \exists R.(D \sqcap E)}$$

New set of rules $\rightsquigarrow$ exponentially many different new axioms

Theorem Instance checking in $\mathcal{ELHI}_\bot$ is P-complete for data and EXP-complete for combined complexity.

Let sat($\mathcal{T}$) be result of applying TBox saturation rules to $\mathcal{T}$.

For each $\mathcal{ELHI}_\bot$ TBox $\mathcal{T}$ and ABox signature Σ define following Datalog program $\Pi(\mathcal{T}, \Sigma)$:

$$
\begin{aligned}
\Pi(\mathcal{T}, \Sigma) = & \{B(x) \leftarrow A_1(x), \ldots, A_n(x) \mid A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathsf{sat}(\mathcal{T})\} \cup \\
& \{B(x) \leftarrow A(y), r(x,y) \mid \exists r.A \sqsubseteq B \in \mathcal{T}\} \cup \\
& \{B(y) \leftarrow A(x), r(x,y) \mid \exists r^-.A \sqsubseteq B \in \mathcal{T}\} \cup \\
& \{s(x,y) \leftarrow r(x,y) \mid r \sqsubseteq s \in \mathsf{sat}(\mathcal{T}), s \in \mathsf{N_R}\} \cup \\
& \{s(y,x) \leftarrow r(x,y) \mid r \sqsubseteq s^- \in \mathsf{sat}(\mathcal{T}), s \in \mathsf{N_R}\} \cup \\
& \{\top(x) \leftarrow A(x) \mid A \in \mathsf{N_C} \cap \Sigma\} \cup \\
& \{\top(x) \leftarrow r(x,y) \mid r \in \mathsf{N_R} \cap \Sigma\} \cup \\
& \{\top(x) \leftarrow r(y,x) \mid r \in \mathsf{N_R} \cap \Sigma\}
\end{aligned}
$$

Theorem For every finite signature Σ and $\mathcal{ELHI}_\bot$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathsf{sig}(\mathcal{A}) \subseteq \Sigma$:

1. $\mathcal{K}$ **is unsatisfiable** iff $\mathsf{ans}((\Pi(\mathcal{T}, \Sigma), \bot), \mathcal{I}_\mathcal{A}) \neq \emptyset$;
2. If $\mathcal{K}$ **is satisfiable**, then for all $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{Ind}(\mathcal{A})$:
   - $\mathcal{K} \models A(a)$ iff $a \in \mathsf{ans}((\Pi(\mathcal{T}, \Sigma), A), \mathcal{I}_\mathcal{A})$;
   - $\mathcal{K} \models r(a, b)$ iff $(a, b) \in \mathsf{ans}((\Pi(\mathcal{T}, \Sigma), r), \mathcal{I}_\mathcal{A})$.

This means:

- get **Datalog rewriting** of instance queries in $\mathcal{ELHI}_\bot$
- can use **Datalog program to create saturated ABox**

SATURATION AS A DATALOG PROGRAM: AN EXAMPLE

The Datalog program associated with our example:

$\mathsf{PastaDish}(x) \leftarrow \mathsf{PenneArrab}(x)$	$\mathsf{PenneArrab} \sqsubseteq \mathsf{PastaDish}$
$\mathsf{Dish}(x) \leftarrow \mathsf{PastaDish}(x)$	$\mathsf{PastaDish} \sqsubseteq \mathsf{Dish}$
$\mathsf{Spicy}(x) \leftarrow \mathsf{Peperonc}(x)$	$\mathsf{Peperonc} \sqsubseteq \mathsf{Spicy}$
$\mathsf{Spicy}(x) \leftarrow \mathsf{hasIngred}(x, y), \mathsf{Spicy}(y)$	$\exists \mathsf{hasIngred}.\mathsf{Spicy} \sqsubseteq \mathsf{Spicy}$
$\mathsf{SpicyDish}(x) \leftarrow \mathsf{Spicy}(x), \mathsf{Dish}(x)$	$\mathsf{Dish} \sqcap \mathsf{Spicy} \sqsubseteq \mathsf{SpicyDish}$
$\mathsf{Spicy}(x) \leftarrow \mathsf{ArrabSauce}(x)$	$\mathsf{ArrabSauce} \sqsubseteq \mathsf{Spicy}$
$\mathsf{Spicy}(x) \leftarrow \mathsf{PenneArrab}(x)$	$\mathsf{PenneArrab} \sqsubseteq \mathsf{Spicy}$
$\mathsf{Dish}(x) \leftarrow \mathsf{PenneArrab}(x)$	$\mathsf{PenneArrab} \sqsubseteq \mathsf{Dish}$
$\mathsf{SpicyDish}(x) \leftarrow \mathsf{PenneArrab}$	$\mathsf{PenneArrab} \sqsubseteq \mathsf{SpicyDish}$

(technically, also have $\mathcal{T}$-independent rules for $\top$...)