Ontologies & Description Logics

KNOWLEDGE REPRESENTATION

Meghyn Bienvenu (LaBRI - *CNRS & Université de Bordeaux*)

REASONING WITH LIGHTWEIGHT DLS

LIGHTWEIGHT ONTOLOGY LANGUAGES

Some applications require very large ontologies and/or data

Scalability concerns led to proposal of DLs with lower complexity

$\mathcal{EL}$ family of DLs (basis for OWL 2 EL)

- designed to allow efficient reasoning with large ontologies
- key technique: saturation ($\sim$ forward chaining)

DL-Lite family of DLs (basis for OWL 2 QL)

- designed for ontology-mediated query answering
- key technique: query rewriting ($\sim$ backward chaining)

REASONING IN $\mathcal{EL}$

THE EL FAMILY: SIMPLER LOGICS FOR SCALABLE REASONING

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

This family of DLs is well suited for biomedical applications.

Examples of large biomedical ontologies:

- GO (Gene Ontology), around 20,000 concepts
- NCI (cancer ontology), around 30,000 concepts
- SNOMED (medical ontology), over 350,000 concepts (!)

$$\mathsf{Pericarditis} \sqsubseteq \mathsf{Inflammation} \sqcap \exists \mathsf{loc}.\mathsf{Pericardium}$$
$$\mathsf{Pericardium} \sqsubseteq \mathsf{Tissue} \sqcap \exists \mathsf{partOf}.\mathsf{Heart} \qquad \mathsf{Inflammation} \sqsubseteq \mathsf{Disease}$$
$$\mathsf{Disease} \sqcap \exists \mathsf{loc}.\exists \mathsf{partOf}.\mathsf{Heart} \sqsubseteq \mathsf{HeartDisease}$$

SYNTAX OF EL

In $\mathcal{EL}$, complex concepts are built as follows:

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists r.C$$

Only concept inclusions $C_1 \sqsubseteq C_2$ in the TBox

SYNTAX OF EL

In $\mathcal{EL}$, complex concepts are built as follows:

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists r.C$$

Only concept inclusions $C_1 \sqsubseteq C_2$ in the TBox

Some possible extensions:

- $\bot$ (to express disjoint classes)
- domain restrictions $\text{dom}(r) \sqsubseteq C$
- range restrictions $\text{range}(r) \sqsubseteq C$
- complex role inclusions $r_1 \circ \ldots \circ r_n \sqsubseteq r_{n+1}$ (transitivity:$r \circ r \sqsubseteq r$)

OWL 2 EL profile includes all these extensions

SYNTAX OF EL

In $\mathcal{EL}$, complex concepts are built as follows:

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists r.C$$

Only concept inclusions $C_1 \sqsubseteq C_2$ in the TBox

Some possible extensions:

- $\bot$ (to express disjoint classes)
- domain restrictions $\text{dom}(r) \sqsubseteq C$
- range restrictions $\text{range}(r) \sqsubseteq C$
- complex role inclusions $r_1 \circ \ldots \circ r_n \sqsubseteq r_{n+1}$ (transitivity:$r \circ r \sqsubseteq r$)

OWL 2 EL profile includes all these extensions

We will focus on plain $\mathcal{EL}$ (without these extensions)

NORMAL FORM FOR EL TBOXES

$\mathcal{T}$ is in **normal form** if it contains only inclusions of the forms:

$$A \sqsubseteq B \quad A_1 \sqcap A_2 \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

where A, A_1, A_2, B are concept names (or $\top$).

Use term **basic axioms** for such inclusions, and **basic assertions** to refer to non-complex assertions

NORMAL FORM FOR EL TBOXES

$\mathcal{T}$ is in **normal form** if it contains only inclusions of the forms:

$$A \sqsubseteq B \quad A_1 \sqcap A_2 \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

where A, A_1, A_2, B are concept names (or $\top$).

Use term **basic axioms** for such inclusions, and **basic assertions** to refer to non-complex assertions

Theorem: for every $\mathcal{EL}$ TBox $\mathcal{T}$, we can **construct in PTIME a TBox $\mathcal{T}'$ in normal form (possibly using new concept names)** such that:

- for every inclusion $C \sqsubseteq D$ which uses only concept names from $\mathcal{T}$, we have $\mathcal{T} \models C \sqsubseteq D$ **iff** $\mathcal{T}' \models C \sqsubseteq D$
- for every ABox $\mathcal{A}$ and assertion α that only uses concept names from $(\mathcal{T}, \mathcal{A})$, we have $\mathcal{T}, \mathcal{A} \models \alpha$ **iff** $\mathcal{T}', \mathcal{A} \models \alpha$

NORMAL FORM FOR EL TBOXES

$\mathcal{T}$ is in **normal form** if it contains only inclusions of the forms:

$$A \sqsubseteq B \quad A_1 \sqcap A_2 \sqsubseteq B \quad A \sqsubseteq \exists r.B \quad \exists r.A \sqsubseteq B$$

where A, A_1, A_2, B are concept names (or $\top$).

Use term **basic axioms** for such inclusions, and **basic assertions** to refer to non-complex assertions

Theorem: for every $\mathcal{EL}$ TBox $\mathcal{T}$, we can **construct in PTIME a TBox $\mathcal{T}'$ in normal form (possibly using new concept names)** such that:

- for every inclusion $C \sqsubseteq D$ which uses only concept names from $\mathcal{T}$, we have $\mathcal{T} \models C \sqsubseteq D$ **iff** $\mathcal{T}' \models C \sqsubseteq D$
- for every ABox $\mathcal{A}$ and assertion α that only uses concept names from $(\mathcal{T}, \mathcal{A})$, we have $\mathcal{T}, \mathcal{A} \models \alpha$ **iff** $\mathcal{T}', \mathcal{A} \models \alpha$

More specifically: $\mathcal{T}'$ is a **conservative extension** of $\mathcal{T}$ (see exercises)

NORMALIZATION PROCEDURE

Exhaustively apply the following **normalization rules** to $\mathcal{T}$: (any order)

$$
\begin{aligned}
\widehat{D} \sqsubseteq \hat{E} &\rightsquigarrow \hat{D} \sqsubseteq A_{\text{new}} \quad A_{\text{new}} \sqsubseteq \hat{E} \\
C \sqcap \widehat{D} \sqsubseteq B &\rightsquigarrow \hat{D} \sqsubseteq A_{\text{new}} \quad C \sqcap A_{\text{new}} \sqsubseteq B \\
\widehat{D} \sqcap C \sqsubseteq B &\rightsquigarrow \hat{D} \sqsubseteq A_{\text{new}} \quad A_{\text{new}} \sqcap C \sqsubseteq B \\
\exists r.\widehat{D} \sqsubseteq B &\rightsquigarrow \hat{D} \sqsubseteq A_{\text{new}} \quad \exists r.A_{\text{new}} \sqsubseteq B \\
B \sqsubseteq \exists r.\widehat{D} &\rightsquigarrow B \sqsubseteq \exists r.A_{\text{new}} \quad A_{\text{new}} \sqsubseteq \hat{D} \\
B \sqsubseteq D \sqcap E &\rightsquigarrow B \sqsubseteq D \quad B \sqsubseteq E
\end{aligned}
$$

where:

- C, D, E are arbitrary $\mathcal{EL}$ concepts,
- $\widehat{D}, \widehat{E}$ are neither concept names nor $\top$,
- B is a concept name,
- A_{new} is a **fresh (new) concept name**

EXAMPLE: NORMALIZATION

Applying the fourth rule to $\exists r.(\exists s.A \sqcap H) \sqsubseteq B \sqcap D$

$$\exists s.A \sqcap H \sqsubseteq \mathbf{E} \qquad \exists r.\mathbf{E} \sqsubseteq B \sqcap D$$

Use third rule to transform $\exists s.A \sqcap H \sqsubseteq E$ into

$$\exists s.A \sqsubseteq \mathbf{F} \qquad \mathbf{F} \sqcap H \sqsubseteq E$$

Last rule used to replace $\exists \mathbf{r}.\mathbf{E} \sqsubseteq \mathbf{B} \sqcap \mathbf{D}$ by

$$\exists r.E \sqsubseteq B \qquad \exists r.E \sqsubseteq D$$

End result:

$$\exists s.A \sqsubseteq F \quad F \sqcap H \sqsubseteq E \quad \exists r.E \sqsubseteq B \quad \exists r.E \sqsubseteq D$$

SATURATION RULES FOR EL

Rules for deriving ontology axioms

$$\frac{}{A \sqsubseteq A}\ \text{T1} \qquad \frac{}{A \sqsubseteq \top}\ \text{T2} \qquad \frac{A \sqsubseteq B \quad B \sqsubseteq D}{A \sqsubseteq D}\ \text{T3}$$

$$\frac{A \sqsubseteq B_1 \quad A \sqsubseteq B_2 \quad B_1 \sqcap B_2 \sqsubseteq D}{A \sqsubseteq D}\ \text{T4} \qquad \frac{A \sqsubseteq \exists r.B_1 \quad B_1 \sqsubseteq B_2 \quad \exists r.B_2 \sqsubseteq D}{A \sqsubseteq D}\ \text{T5}$$

Rules for deriving assertions

$$\frac{A \sqsubseteq B \quad A(c)}{B(c)}\ \text{A1} \qquad \frac{A_1 \sqcap A_2 \sqsubseteq B \quad A_1(c) \quad A_2(c)}{B(c)}\ \text{A2} \qquad \frac{\exists r.A \sqsubseteq B \quad r(c,d) \quad A(d)}{B(c)}\ \text{A3}$$

Premises = axioms / assertions above the line
Conclusion = axiom / assertion below the line

SATURATION PROCEDURE

Assume w.l.o.g. that start from KB whose TBox is in normal form & whose ABox contains $\top(a)$ for each of its individuals a

SATURATION PROCEDURE

Assume w.l.o.g. that start from KB whose **TBox is in normal form** & whose **ABox contains $\top(a)$ for each of its individuals a**

Instantiated rule:

- obtained from one of the 'abstract' saturation rules by replacing A, B, D by $\mathcal{EL}$-concepts and r by some role name
- must **only contain basic axioms & assertions** (important!)

SATURATION PROCEDURE

Assume w.l.o.g. that start from KB whose **TBox is in normal form** & whose **ABox contains $\top(a)$ for each of its individuals a**

Instantiated rule:

- obtained from one of the 'abstract' saturation rules by replacing A, B, D by $\mathcal{EL}$-concepts and r by some role name
- must **only contain basic axioms & assertions** (important!)

Instantiated rule with premises $\alpha_1, \ldots, \alpha_n$ and conclusion β is **applicable in $\mathcal{K}$** if $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathcal{K}$ and $\beta \notin \mathcal{K}$

- in this case, can **apply the rule by adding β to $\mathcal{K}$**

Saturation procedure: **exhaustively apply instantiated rules** until no rule is applicable

EXAMPLE: SATURATION RULES

TBox $\mathcal{T}$ contains axioms:

(1) $\exists \mathsf{hasIngred}.\mathsf{Spicy} \sqsubseteq \mathsf{Spicy}$ (2) $\mathsf{Spicy} \sqcap \mathsf{Dish} \sqsubseteq \mathsf{SpicyDish}$

(3) $\mathsf{ArrabSauce} \sqsubseteq \exists \mathsf{hasIngred}.\mathsf{Chili}$ (4) $\mathsf{Chili} \sqsubseteq \mathsf{Spicy}$

ABox $\mathcal{A}$ contains:

(5) $\mathsf{Dish}(p)$ (6) $\mathsf{hasIngred}(p, s)$ (7) $\mathsf{ArrabSauce}(s)$

EXAMPLE: SATURATION RULES

TBox $\mathcal{T}$ contains axioms:

(1) $\exists \text{hasIngred}.\text{Spicy} \sqsubseteq \text{Spicy}$ (2) $\text{Spicy} \sqcap \text{Dish} \sqsubseteq \text{SpicyDish}$

(3) $\text{ArrabSauce} \sqsubseteq \exists \text{hasIngred}.\text{Chili}$ (4) $\text{Chili} \sqsubseteq \text{Spicy}$

ABox $\mathcal{A}$ contains:

(5) $\text{Dish}(p)$ (6) $\text{hasIngred}(p, s)$ (7) $\text{ArrabSauce}(s)$

Saturation procedure adds the following axioms and assertions:

(8) $\text{ArrabSauce} \sqsubseteq \text{Spicy}$	using (1), (3), (4) and rule T5
(9) $\text{Spicy}(s)$	using (7), (8), and rule A1
(10) $\text{Spicy}(p)$	using (1), (6), (9), and rule A3
(11) $\text{SpicyDish}(p)$	using (2), (5), (10), and rule A2

EXAMPLE: SATURATION RULES

TBox $\mathcal{T}$ contains axioms:

(1) $\exists \text{hasIngred}.\text{Spicy} \sqsubseteq \text{Spicy}$ (2) $\text{Spicy} \sqcap \text{Dish} \sqsubseteq \text{SpicyDish}$

(3) $\text{ArrabSauce} \sqsubseteq \exists \text{hasIngred}.\text{Chili}$ (4) $\text{Chili} \sqsubseteq \text{Spicy}$

ABox $\mathcal{A}$ contains:

(5) $\text{Dish}(p)$ (6) $\text{hasIngred}(p, s)$ (7) $\text{ArrabSauce}(s)$

Saturation procedure adds the following axioms and assertions:

(8) $\text{ArrabSauce} \sqsubseteq \text{Spicy}$	using (1), (3), (4) and rule T5
(9) $\text{Spicy}(s)$	using (7), (8), and rule A1
(10) $\text{Spicy}(p)$	using (1), (6), (9), and rule A3
(11) $\text{SpicyDish}(p)$	using (2), (5), (10), and rule A2

Examining the result, return p as answer to instance query $q(x) = \text{SpicyDish}(x)$

USING SATURATED KB FOR REASONING

Denote by $sat(\mathcal{K})$ or $sat(\mathcal{T}, \mathcal{A})$ (resp. $sat(\mathcal{T})$) result of exhaustively applying saturation rules to KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (resp. TBox $\mathcal{T}$)

USING SATURATED KB FOR REASONING

Denote by $sat(\mathcal{K})$ or $sat(\mathcal{T}, \mathcal{A})$ (resp. $sat(\mathcal{T})$) result of exhaustively applying saturation rules to KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (resp. TBox $\mathcal{T}$)

To find all **instances of concept name** A w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

1. Normalize $\mathcal{T}$, yielding $\mathcal{T}'$, then construct $sat(\mathcal{T}', \mathcal{A})$
2. Return all individuals c such that $A(c) \in sat(\mathcal{T}', \mathcal{A})$.

USING SATURATED KB FOR REASONING

Denote by $sat(\mathcal{K})$ or $sat(\mathcal{T}, \mathcal{A})$ (resp. $sat(\mathcal{T})$) result of exhaustively applying saturation rules to KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (resp. TBox $\mathcal{T}$)

To find all **instances of concept name** A w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

1. Normalize $\mathcal{T}$, yielding $\mathcal{T}'$, then construct $sat(\mathcal{T}', \mathcal{A})$
2. Return all individuals c such that $A(c) \in sat(\mathcal{T}', \mathcal{A})$.

To **test whether** $\mathcal{T} \models A \sqsubseteq B$ (A, B concept names):

1. Normalize $\mathcal{T}$, yielding $\mathcal{T}'$, then construct $sat(\mathcal{T}')$
   (can alternatively construct $sat(\mathcal{T}', \mathcal{A})$ *if have an ABox* $\mathcal{A}$*)*
2. Check if $sat(\mathcal{T}')$ contains $A \sqsubseteq B$, return yes if so, else no.

USING SATURATED KB FOR REASONING

Denote by $sat(\mathcal{K})$ or $sat(\mathcal{T}, \mathcal{A})$ (resp. $sat(\mathcal{T})$) result of exhaustively applying saturation rules to KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (resp. TBox $\mathcal{T}$)

To find all **instances of concept name** A w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

1. Normalize $\mathcal{T}$, yielding $\mathcal{T}'$, then construct $sat(\mathcal{T}', \mathcal{A})$
2. Return all individuals c such that $A(c) \in sat(\mathcal{T}', \mathcal{A})$.

To **test whether** $\mathcal{T} \models A \sqsubseteq B$ (A, B concept names):

1. Normalize $\mathcal{T}$, yielding $\mathcal{T}'$, then construct $sat(\mathcal{T}')$
   (can alternatively construct $sat(\mathcal{T}', \mathcal{A})$ if have an ABox $\mathcal{A}$)
2. Check if $sat(\mathcal{T}')$ contains $A \sqsubseteq B$, return yes if so, else no.

What about **assertions / inclusions involving complex concepts**?

- Use new concept names to represent complex concepts, e.g. if C is a complex concept, add $X_C \sqsubseteq C$ and $C \sqsubseteq X_C$ to $\mathcal{T}$ (with X_C fresh).
- Proceed as above but use X_C in place of C.

PROPERTIES OF SATURATION PROCEDURE

Theorem. All exhaustive sequences of rule applications lead to a unique saturated KB.

Theorem. The saturated KB $sat(\mathcal{K})$ can be **constructed in polynomial time** in $|\mathcal{K}|$

PROPERTIES OF SATURATION PROCEDURE

Theorem. All exhaustive sequences of rule applications lead to a unique saturated KB.

Theorem. The saturated KB $sat(\mathcal{K})$ can be **constructed in polynomial time** in $|\mathcal{K}|$

Theorem. The saturation procedure is **correct and complete for axiom entailment and instance checking** involving concept names. Specifically:

- for every concept inclusion $A \sqsubseteq B$ (with A, B concept names): $\mathcal{T} \models A \sqsubseteq B$ **iff** $A \sqsubseteq B \in sat(\mathcal{K})$ **iff** $A \sqsubseteq B \in sat(\mathcal{T})$

PROPERTIES OF SATURATION PROCEDURE

Theorem. All exhaustive sequences of rule applications lead to a unique saturated KB.

Theorem. The saturated KB $sat(\mathcal{K})$ can be **constructed in polynomial time** in $|\mathcal{K}|$

Theorem. The saturation procedure is **correct and complete for axiom entailment and instance checking** involving concept names. Specifically:

- for every concept inclusion $A \sqsubseteq B$ (with A, B concept names): $\mathcal{T} \models A \sqsubseteq B$ **iff** $A \sqsubseteq B \in sat(\mathcal{K})$ **iff** $A \sqsubseteq B \in sat(\mathcal{T})$
- for every ABox assertion $\alpha = A(b)$ with A a concept name: $\mathcal{K} \models A(b)$ **iff** $A(b) \in sat(\mathcal{K})$

Next slides: sketch proofs for correctness and completeness

CORRECTNESS OF SATURATION

Aim to show that:

- if $A \sqsubseteq B \in sat(\mathcal{K})$, then $\mathcal{T} \models A \sqsubseteq B$
- if $A(b) \in sat(\mathcal{K})$, then $\mathcal{K} \models A(b)$

CORRECTNESS OF SATURATION

Aim to show that:

- if $A \sqsubseteq B \in sat(\mathcal{K})$, then $\mathcal{T} \models A \sqsubseteq B$
- if $A(b) \in sat(\mathcal{K})$, then $\mathcal{K} \models A(b)$

As $sat(\mathcal{K})$ is the result of a sequence of rule applications, it suffices to show the following lemma:

Lemma. If a saturation rule application produces β from the premises $\alpha_1, \ldots, \alpha_n$, and $\mathcal{K} \models \alpha_i$ $(1 \leq i \leq n)$, then $\mathcal{K} \models \beta$.

CORRECTNESS OF SATURATION

Aim to show that:

- if $A \sqsubseteq B \in sat(\mathcal{K})$, then $\mathcal{T} \models A \sqsubseteq B$
- if $A(b) \in sat(\mathcal{K})$, then $\mathcal{K} \models A(b)$

As $sat(\mathcal{K})$ is the result of a sequence of rule applications, it suffices to show the following lemma:

Lemma. If a saturation rule application produces β from the premises $\alpha_1, \ldots, \alpha_n$, and $\mathcal{K} \models \alpha_i$ $(1 \leq i \leq n)$, then $\mathcal{K} \models \beta$.

Proof sketch:

- trivial for rules **T1** and **T2** (produced axioms hold in any model)
- easy arguments for other rules, e.g. for **T3**:
  - suppose $\mathcal{K} \models A \sqsubseteq B$ and $\mathcal{K} \models B \sqsubseteq D$, take any model $\mathcal{I}$ of $\mathcal{K}$ and $e \in A^{\mathcal{I}}$, must have $e \in B^{\mathcal{I}}$ due to $A \sqsubseteq B$, hence $e \in D^{\mathcal{I}}$ due to $B \sqsubseteq D$, yielding $\mathcal{I} \models A \sqsubseteq D$ as required

We prove the contrapositive, namely:

- if $A \sqsubseteq B \notin sat(\mathcal{K})$, then $\mathcal{T} \not\models A \sqsubseteq B$
- if $A(b) \notin sat(\mathcal{K})$, then $\mathcal{K} \not\models A(b)$

We prove the contrapositive, namely:

- if $A \sqsubseteq B \notin sat(\mathcal{K})$, then $\mathcal{T} \not\models A \sqsubseteq B$
- if $A(b) \notin sat(\mathcal{K})$, then $\mathcal{K} \not\models A(b)$

Proof strategy:

- build an interpretation $\mathcal{C}_\mathcal{K}$ from $sat(\mathcal{K})$
- show that $\mathcal{C}_\mathcal{K}$ is a model of $\mathcal{K}$
- show that $\mathcal{C}_\mathcal{K} \not\models A \sqsubseteq B$ when $A \sqsubseteq B \notin sat(\mathcal{K})$
- show that $\mathcal{C}_\mathcal{K} \not\models A(b)$ when $A(b) \notin sat(\mathcal{K})$

Define $\mathcal{C}_\mathcal{K}$, as follows:

- $\Delta^{\mathcal{C}_\mathcal{K}} = \mathsf{Ind}(\mathcal{A}) \cup \{w_A \mid A \text{ concept name appearing in } \mathcal{K}\} \cup \{w_\top\}$
- $A^{\mathcal{C}_\mathcal{K}} = \{b \mid A(b) \in sat(\mathcal{K})\} \cup \{w_B \mid B \sqsubseteq A \in sat(\mathcal{K})\}$
- $r^{\mathcal{C}_\mathcal{K}} = \{(a,b) \mid r(a,b) \in \mathcal{K}\} \cup \{(w_A, w_B) \mid A \sqsubseteq \exists r.B \in sat(\mathcal{K})\}$
  $\cup \{(a, w_B) \mid A \sqsubseteq \exists r.B \in sat(\mathcal{K}), A(a) \in sat(\mathcal{K}) \text{ for some } A\}$
- $a^{\mathcal{C}_\mathcal{K}} = a$ for all $a \in \mathsf{Ind}(\mathcal{A})$

where $\mathsf{Ind}(\mathcal{A})$ is set of individual names in $\mathcal{A}$

Define $\mathcal{C}_\mathcal{K}$, as follows:

- $\Delta^{\mathcal{C}_\mathcal{K}} = \mathsf{Ind}(\mathcal{A}) \cup \{w_A \mid A \text{ concept name appearing in } \mathcal{K}\} \cup \{w_\top\}$
- $A^{\mathcal{C}_\mathcal{K}} = \{b \mid A(b) \in sat(\mathcal{K})\} \cup \{w_B \mid B \sqsubseteq A \in sat(\mathcal{K})\}$
- $r^{\mathcal{C}_\mathcal{K}} = \{(a,b) \mid r(a,b) \in \mathcal{K}\} \cup \{(w_A, w_B) \mid A \sqsubseteq \exists r.B \in sat(\mathcal{K})\}$
  $\cup \{(a, w_B) \mid A \sqsubseteq \exists r.B \in sat(\mathcal{K}), A(a) \in sat(\mathcal{K}) \text{ for some } A\}$
- $a^{\mathcal{C}_\mathcal{K}} = a$ for all $a \in \mathsf{Ind}(\mathcal{A})$

where $\mathsf{Ind}(\mathcal{A})$ is set of individual names in $\mathcal{A}$

Observe that by construction, we have:

- $\mathcal{C}_\mathcal{K} \not\models A \sqsubseteq B$ when $A \sqsubseteq B \notin sat(\mathcal{K})$ $\quad w_A \in A^{\mathcal{C}_\mathcal{K}}$ but $w_A \notin B^{\mathcal{C}_\mathcal{K}}$
- $\mathcal{C}_\mathcal{K} \not\models A(b)$ when $A(b) \notin sat(\mathcal{K})$

for all concept names A, B and individuals b occurring in $\mathcal{K}$

By definition, $\mathcal{C}_{\mathcal{K}}$ is a model of $\mathcal{A}$

By definition, $\mathcal{C}_\mathcal{K}$ is a model of $\mathcal{A}$

To show it is a model of $\mathcal{T}$, consider different kinds of axioms:

- **Case 1:** $A \sqsubseteq B \in \mathcal{T}$ **and** $e \in A^{\mathcal{C}_\mathcal{K}}$
  - If $e \in \mathrm{Ind}(\mathcal{A})$, then $A(e) \in sat(\mathcal{K})$. Due to **A1**, $B(e) \in sat(\mathcal{K})$, so $e \in B^{\mathcal{C}_\mathcal{K}}$.
  - If $e = w_D$, then $D \sqsubseteq A \in sat(\mathcal{T})$. Due to **T3**, $D \sqsubseteq B \in sat(\mathcal{T})$, so $e \in B^{\mathcal{C}_\mathcal{K}}$.

By definition, $\mathcal{C}_\mathcal{K}$ is a model of $\mathcal{A}$

To show it is a model of $\mathcal{T}$, consider different kinds of axioms:

- **Case 1:** $A \sqsubseteq B \in \mathcal{T}$ **and** $e \in A^{\mathcal{C}_\mathcal{K}}$
  - If $e \in \mathrm{Ind}(\mathcal{A})$, then $A(e) \in sat(\mathcal{K})$. Due to **A1**, $B(e) \in sat(\mathcal{K})$, so $e \in B^{\mathcal{C}_\mathcal{K}}$.
  - If $e = w_D$, then $D \sqsubseteq A \in sat(\mathcal{T})$. Due to **T3**, $D \sqsubseteq B \in sat(\mathcal{T})$, so $e \in B^{\mathcal{C}_\mathcal{K}}$.
- **Case 2:** $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ **and** $e \in (A_1 \sqcap A_2)^{\mathcal{C}_\mathcal{K}}$
  - similar argument using **A2** and **T4**
- **Case 3:** $A \sqsubseteq \exists r.B \in \mathcal{T}$ **and** $e \in A^{\mathcal{C}_\mathcal{K}}$
  - argument uses **T3**
- **Case 4:** $\exists r.A \sqsubseteq B$, $e' \in A^{\mathcal{C}_\mathcal{K}}$, **and** $(e, e') \in r^{\mathcal{C}_\mathcal{K}}$
  - argument uses **A3** and **T5**

By definition, $\mathcal{C}_\mathcal{K}$ is a model of $\mathcal{A}$

To show it is a model of $\mathcal{T}$, consider different kinds of axioms:

- **Case 1:** $A \sqsubseteq B \in \mathcal{T}$ **and** $e \in A^{\mathcal{C}_\mathcal{K}}$
  - If $e \in \mathrm{Ind}(\mathcal{A})$, then $A(e) \in sat(\mathcal{K})$. Due to **A1**, $B(e) \in sat(\mathcal{K})$, so $e \in B^{\mathcal{C}_\mathcal{K}}$.
  - If $e = w_D$, then $D \sqsubseteq A \in sat(\mathcal{T})$. Due to **T3**, $D \sqsubseteq B \in sat(\mathcal{T})$, so $e \in B^{\mathcal{C}_\mathcal{K}}$.
- **Case 2:** $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ **and** $e \in (A_1 \sqcap A_2)^{\mathcal{C}_\mathcal{K}}$
  - similar argument using **A2** and **T4**
- **Case 3:** $A \sqsubseteq \exists r.B \in \mathcal{T}$ **and** $e \in A^{\mathcal{C}_\mathcal{K}}$
  - argument uses **T3**
- **Case 4:** $\exists r.A \sqsubseteq B$, $e' \in A^{\mathcal{C}_\mathcal{K}}$, **and** $(e, e') \in r^{\mathcal{C}_\mathcal{K}}$
  - argument uses **A3** and **T5**

Call $\mathcal{C}_\mathcal{K}$ the (compact) canonical model for $\mathcal{K}$

COMPLEXITY OF REASONING IN EL

Theorem. Axiom entailment and instance checking over $\mathcal{EL}$ KBs are PTIME-complete

- upper bound: saturation procedure from previous slides
- lower bound: entailment from propositional Horn theories

COMPLEXITY OF REASONING IN EL

Theorem. Axiom entailment and instance checking over $\mathcal{EL}$ KBs are PTIME-complete

- upper bound: saturation procedure from previous slides
- lower bound: entailment from propositional Horn theories

Note: with only $\sqcap$ and $\forall r.C$, same problems are EXPTIME-complete!

COMPLEXITY OF REASONING IN EL

Theorem. Axiom entailment and instance checking over $\mathcal{EL}$ KBs are PTIME-complete

- upper bound: saturation procedure from previous slides
- lower bound: entailment from propositional Horn theories

Note: with only $\sqcap$ and $\forall r.C$, same problems are EXPTIME-complete!

Further advantage of saturation approach: 'single-pass' reasoning

- compute saturation once, then read off all entailed assertions and inclusions involving concept names

COMPLEXITY OF REASONING IN EL

Theorem. Axiom entailment and instance checking over $\mathcal{EL}$ KBs are PTIME-complete

- upper bound: saturation procedure from previous slides
- lower bound: entailment from propositional Horn theories

Note: with only $\sqcap$ and $\forall r.C$, same problems are EXPTIME-complete!

Further advantage of saturation approach: 'single-pass' reasoning

- compute saturation once, then read off all entailed assertions and inclusions involving concept names

In practice:

- huge ontologies like SNOMED can be classified in a few seconds

EXTENSIONS OF EL

We can add all of the following without losing tractability:

- $\bot$
- $\mathrm{dom}(r) \sqsubseteq C$, $\mathrm{range}(r) \sqsubseteq C$
- $r_1 \circ \ldots \circ r_n \sqsubseteq r_{n+1}$ (complex role inclusions)

EXTENSIONS OF EL

We can add all of the following without losing tractability:

- $\bot$
- $\mathrm{dom}(r) \sqsubseteq C$, $\mathrm{range}(r) \sqsubseteq C$
- $r_1 \circ \ldots \circ r_n \sqsubseteq r_{n+1}$ (complex role inclusions)

But adding any of the following makes reasoning EXPTIME-hard:

- negation $\neg$
- disjunction $\sqcup$
- at-least or at-most restrictions: $\geq 2r$, $\leq 1r$
- functional roles $(\mathrm{funct}\, r)$
- inverse roles r^-

GLIMPSE AT ELI

The DL $\mathcal{ELI}$ is obtained by adding inverse roles to $\mathcal{EL}$

Reasoning in $\mathcal{ELI}$ is much more difficult (EXPTIME-complete)

However, $\mathcal{ELI}$ retains some nice properties:

- admits a canonical model, hence no 'case-based' reasoning

Can extend saturation procedure to $\mathcal{ELI}$

- still deterministic
- may be exponential since need to consider sets of concept names
  - deduce $A \sqcap D \sqsubseteq \exists r.(B \sqcap D)$ from $A \sqsubseteq \exists r.B$ and $\exists r^{-}.D \sqsubseteq E$

In practice: $\mathcal{ELI}$ and other 'Horn DLs' easier to handle than $\mathcal{ALC}$

REASONING IN DL-LITE

USING ONTOLOGIES TO ACCESS DATA

Aim: enrich databases (DBs) with ontologies

- convenient vocabulary for users to specify queries
- link multiple datasets with different schemas
- knowledge in ontology can yield additional answers to queries

USING ONTOLOGIES TO ACCESS DATA

Aim: enrich databases (DBs) with ontologies

- convenient vocabulary for users to specify queries
- link multiple datasets with different schemas
- knowledge in ontology can yield additional answers to queries

Desiderata:

- efficiency is crucial – must scale up to huge datasets
- instance queries rather simple – want more expressive queries like in databases
  - conjunctive queries (CQs) $\sim$ select-project-join queries in SQL

USING ONTOLOGIES TO ACCESS DATA

Aim: enrich databases (DBs) with ontologies

- convenient vocabulary for users to specify queries
- link multiple datasets with different schemas
- knowledge in ontology can yield additional answers to queries

Desiderata:

- efficiency is crucial – must scale up to huge datasets
- instance queries rather simple – want more expressive queries like in databases
  - conjunctive queries (CQs) $\sim$ select-project-join queries in SQL

DL-Lite family: designed for efficient conjunctive query answering

DL-LITE

We consider the **dialect DL-Lite$_R$** (basis for **OWL 2 QL profile**)

DL-Lite$_R$ axioms:

- **concept inclusions** $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$
- **role inclusions** $S_1 \sqsubseteq S_2$, $S_1 \sqsubseteq \neg S_2$

where $B := A \mid \exists S$ $S := r \mid r^-$

Example axioms:

- Every professor teaches something: $\text{Prof} \sqsubseteq \exists \text{teaches}$
- Everything that is taught is a course: $\exists \text{teaches}^- \sqsubseteq \text{Course}$
- Director of dept implies member of dept: $\text{directorOf} \sqsubseteq \text{memberOf}$

CONJUNCTIVE QUERIES

An **atom** takes the form $A(t_1)$ or $r(t_1, t_2)$ where:

- A is a concept name, r a role name
- each t_i is either a variable or individual name

A **conjunctive query (CQ)** has the form

$$q(x_1, \ldots, x_k) \quad = \quad \exists y_1, \ldots, y_m \;\; \alpha_1 \wedge \ldots \wedge \alpha_n$$

where each α_i is an atom with variables drawn from $x_1, \ldots, x_k, y_1, \ldots, y_m$.

- $y_1, \ldots, y_m$ are called **quantified / existential variables**
- $x_1, \ldots, x_k$ are called **answer variables**

Conjunctive queries correspond to **select-project-join queries in SQL**

SEMANTICS OF CONJUNCTIVE QUERIES (1)

Boolean CQ = CQ that has no answer variables

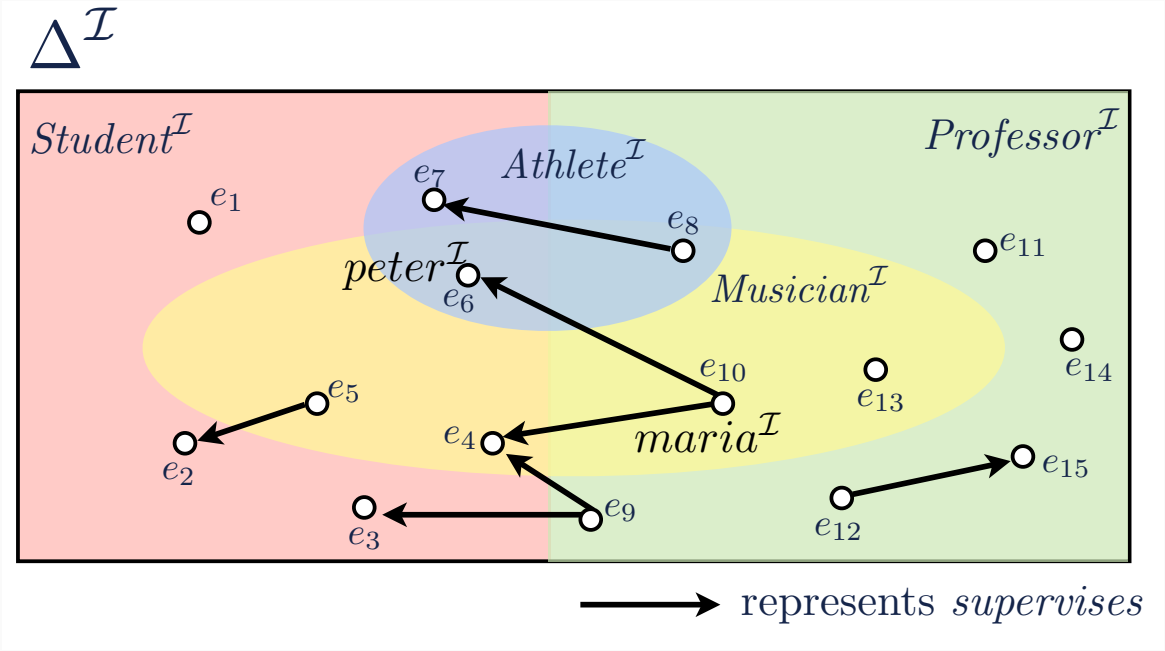
Satisfaction of a Boolean CQ in an interpretation:

Interpretation $\mathcal{I}$ **satisfies a Boolean CQ** q if there exists a **function** π **mapping each term of** q **to an element of** $\Delta^{\mathcal{I}}$ such that:

- for every individual a in q: $\pi(a) = a^{\mathcal{I}}$
- for every atom $A(t) \in q$: $\pi(t) \in A^{\mathcal{I}}$
- for every atom $r(t_1, t_2) \in q$: $(\pi(t_1), \pi(t_2)) \in r^{\mathcal{I}}$

EXAMPLE: SATISFACTION IN AN INTERPRETATION

Reconsider the example interpretation $\mathcal{I}$:



Which of the following Boolean CQs are satisfied in $\mathcal{I}$?

(1) $supervises(maria, peter)$

(2) $\exists x\, supervises(x, peter) \wedge Student(x)$

(3) $\exists x, y\, Musician(x) \wedge supervises(x, y) \wedge Athlete(y)$

(4) $\exists x, y, z\, supervises(x, y) \wedge supervises(y, z)$

SEMANTICS OF CONJUNCTIVE QUERIES (2)

Entailment of a Boolean CQ:

Boolean CQ q **is entailed from** $\mathcal{K}$ (written $\mathcal{K} \models q$)

if and only if **every model of** $\mathcal{K}$ **satisfies** q.

Certain answers to a CQ:

Suppose q has answer variables $x_1, \ldots, x_k$. A tuple $\vec{a} = (a_1, \ldots, a_k)$ of individuals from $\mathcal{A}$ is a **(certain) answer to** q **w.r.t.** $\mathcal{K}$ if and only if

$$\mathcal{K} \models q(\vec{a})$$

where $q(\vec{a})$ is the Boolean CQ q with every x_i replaced by a_i.

We denote by $\mathrm{cert}(q, \mathcal{K})$ the certain answers to q w.r.t. $\mathcal{K}$

EXAMPLE: CERTAIN ANSWERS

DL-Lite ontology:

$$\begin{array}{lll} \mathsf{Prof} \sqsubseteq \mathsf{Faculty} & \mathsf{Researcher} \sqsubseteq \mathsf{Faculty} & \mathsf{Faculty} \sqsubseteq \neg \mathsf{Course} \\ \mathsf{Prof} \sqsubseteq \exists \mathsf{teaches} & \exists \mathsf{teaches}^{-} \sqsubseteq \mathsf{Course} & \end{array}$$

ABox:

$$\mathcal{D} = \{\mathsf{Prof}(\mathsf{anna}), \mathsf{Researcher}(\mathsf{tom}), \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})\}$$

Conjunctive query: $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

EXAMPLE: CERTAIN ANSWERS

DL-Lite ontology:

$$\begin{array}{lll} \mathsf{Prof} \sqsubseteq \mathsf{Faculty} & \mathsf{Researcher} \sqsubseteq \mathsf{Faculty} & \mathsf{Faculty} \sqsubseteq \neg \mathsf{Course} \\ \mathsf{Prof} \sqsubseteq \exists \mathsf{teaches} & \exists \mathsf{teaches}^{-} \sqsubseteq \mathsf{Course} & \end{array}$$

ABox:

$$\mathcal{D} = \{\mathsf{Prof}(\mathsf{anna}), \mathsf{Researcher}(\mathsf{tom}), \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})\}$$

Conjunctive query: $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

Get the following certain answers:

- anna — $\mathsf{Prof}(\mathsf{anna}) + \mathsf{Prof} \sqsubseteq \mathsf{Faculty} + \mathsf{Prof} \sqsubseteq \exists \mathsf{teaches}$
- tom — $\mathsf{Researcher}(\mathsf{tom}) + \mathsf{Researcher} \sqsubseteq \mathsf{Faculty} + \mathsf{teaches}(\mathsf{tom}, \mathsf{cs101})$

QUERY REWRITING

Idea: reduce to standard database (DB) query evaluation

- rewriting step: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ first-order (SQL) query q'
- evaluation step: evaluate query q' using relational DB system

Advantage: harness efficiency of relational database systems

QUERY REWRITING

Idea: reduce to standard database (DB) query evaluation

- rewriting step: TBox $\mathcal{T}$ + query $q \rightsquigarrow$ first-order (SQL) query q'
- evaluation step: evaluate query q' using relational DB system

Advantage: harness efficiency of relational database systems

Key notion: first-order (FO) rewriting

- FO query q' is an FO-rewriting of q w.r.t. $\mathcal{T}$ iff for every ABox $\mathcal{A}$:

$$\mathcal{T}, \mathcal{A} \models q(\vec{a}) \qquad \Leftrightarrow \qquad \mathcal{I}_{\mathcal{A}} \models q'(\vec{a})$$

where $\mathcal{I}_{\mathcal{A}}$ is the interpretation based upon $\mathcal{A}$, defined by setting $\Delta^{\mathcal{I}} = \mathsf{Ind}(\mathcal{A})$, $A^{\mathcal{I}} = \{c \mid A(c) \in \mathcal{A}\}$, $r^{\mathcal{I}} = \{(c,d) \mid r(c,d) \in \mathcal{A}\}$.

In words: evaluating q' over $\mathcal{A}$ (viewed as DB) yields certain answers

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite ontology $\mathcal{T}$:

$$\begin{array}{lll} \mathsf{Prof} \sqsubseteq \mathsf{Faculty} & \mathsf{Researcher} \sqsubseteq \mathsf{Faculty} & \mathsf{Faculty} \sqsubseteq \neg \mathsf{Course} \\ \mathsf{Prof} \sqsubseteq \exists \mathsf{teaches} & \exists \mathsf{teaches}^{-} \sqsubseteq \mathsf{Course} & \end{array}$$

and the query $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite ontology $\mathcal{T}$:

$$\begin{array}{lll} \mathsf{Prof} \sqsubseteq \mathsf{Faculty} & \mathsf{Researcher} \sqsubseteq \mathsf{Faculty} & \mathsf{Faculty} \sqsubseteq \neg \mathsf{Course} \\ \mathsf{Prof} \sqsubseteq \exists \mathsf{teaches} & \exists \mathsf{teaches}^{-} \sqsubseteq \mathsf{Course} & \end{array}$$

and the query $q(x) = \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y)$

The following query is a rewriting of $q(x)$ w.r.t. $\mathcal{T}$:

$$q(x) \quad \vee \quad \mathsf{Prof}(x) \quad \vee \quad \exists y.\mathsf{Researcher}(x) \wedge \mathsf{teaches}(x, y)$$

EXAMPLE: QUERY REWRITING IN DL-LITE

Reconsider the DL-Lite ontology $\mathcal{T}$:

$$\begin{array}{lll} \text{Prof} \sqsubseteq \text{Faculty} & \text{Researcher} \sqsubseteq \text{Faculty} & \text{Faculty} \sqsubseteq \neg\text{Course} \\ \text{Prof} \sqsubseteq \exists\text{teaches} & \exists\text{teaches}^- \sqsubseteq \text{Course} & \end{array}$$

and the query $q(x) = \exists y.\text{Faculty}(x) \wedge \text{teaches}(x, y)$

The following query is a rewriting of $q(x)$ w.r.t. $\mathcal{T}$:

$$q(x) \quad \vee \quad \text{Prof}(x) \quad \vee \quad \exists y.\text{Researcher}(x) \wedge \text{teaches}(x, y)$$

Evaluating the rewritten query over the earlier dataset

$$\{\text{Prof}(\text{anna}), \text{Researcher}(\text{tom}), \text{teaches}(\text{tom}, \text{cs101})\}$$

produces the two certain answers: anna and tom

APPLICABLE AXIOMS

Now we consider how to compute rewritings.

Idea: apply positive inclusions (PIs) in TBox from right to left

First, we define when this is possible.

A PI I is **applicable to an atom** $A(x)$ if it has A in its right-hand side.

A PI I is **applicable to an atom** $r(x_1, x_2)$ if:

- $x_2 = \_$ and the right-hand side of I is $\exists r$, or
- $x_1 = \_$ and the right-hand side of I is $\exists r^-$, or
- I is a role inclusion and its right-hand side is either r or r^-.

REWRITING ALGORITHM: ATOMS

Let I be an inclusion that is applicable to atom α.

The **rewriting** $\mathrm{ra}(\alpha, I)$ **of atom** α **using inclusion** I is as follows:

- if $\alpha = A(x)$ and $I = B \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = B(x)$
- if $\alpha = A(x)$ and $I = \exists r \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(x, \_)$
- if $\alpha = A(x)$ and $I = \exists r^- \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(\_, x)$

REWRITING ALGORITHM: ATOMS

Let I be an inclusion that is applicable to atom α.

The **rewriting** $\mathrm{ra}(\alpha, I)$ **of atom** α **using inclusion** I is as follows:

- if $\alpha = A(x)$ and $I = B \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = B(x)$
- if $\alpha = A(x)$ and $I = \exists r \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(x, \_)$
- if $\alpha = A(x)$ and $I = \exists r^- \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(\_, x)$
- if $\alpha = r(x, \_)$ and $I = A \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = A(x)$
- if $\alpha = r(x, \_)$ and $I = \exists s \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = s(x, \_)$
- if $\alpha = r(x, \_)$ and $I = \exists s^- \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = s(\_, x)$

REWRITING ALGORITHM: ATOMS

Let I be an inclusion that is applicable to atom α.

The **rewriting** $\mathrm{ra}(\alpha, I)$ **of atom** α **using inclusion** I is as follows:

- if $\alpha = A(x)$ and $I = B \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = B(x)$
- if $\alpha = A(x)$ and $I = \exists r \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(x, \_)$
- if $\alpha = A(x)$ and $I = \exists r^- \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(\_, x)$
- if $\alpha = r(x, \_)$ and $I = A \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = A(x)$
- if $\alpha = r(x, \_)$ and $I = \exists s \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = s(x, \_)$
- if $\alpha = r(x, \_)$ and $I = \exists s^- \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = s(\_, x)$
- if $\alpha = r(\_, x)$ and $I = A \sqsubseteq \exists r^-$, then $\mathrm{ra}(\alpha, I) = A(x)$
- if $\alpha = r(\_, x)$ and $I = \exists s \sqsubseteq \exists r^-$, then $\mathrm{ra}(\alpha, I) = s(x, \_)$
- if $\alpha = r(\_, x)$ and $I = \exists s^- \sqsubseteq \exists r^-$, then $\mathrm{ra}(\alpha, I) = s(\_, x)$

REWRITING ALGORITHM: ATOMS

Let I be an inclusion that is applicable to atom α.

The **rewriting** $\mathrm{ra}(\alpha, I)$ **of atom** α **using inclusion** I is as follows:

- if $\alpha = A(x)$ and $I = B \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = B(x)$
- if $\alpha = A(x)$ and $I = \exists r \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(x, \_)$
- if $\alpha = A(x)$ and $I = \exists r^- \sqsubseteq A$, then $\mathrm{ra}(\alpha, I) = r(\_, x)$
- if $\alpha = r(x, \_)$ and $I = A \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = A(x)$
- if $\alpha = r(x, \_)$ and $I = \exists s \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = s(x, \_)$
- if $\alpha = r(x, \_)$ and $I = \exists s^- \sqsubseteq \exists r$, then $\mathrm{ra}(\alpha, I) = s(\_, x)$
- if $\alpha = r(\_, x)$ and $I = A \sqsubseteq \exists r^-$, then $\mathrm{ra}(\alpha, I) = A(x)$
- if $\alpha = r(\_, x)$ and $I = \exists s \sqsubseteq \exists r^-$, then $\mathrm{ra}(\alpha, I) = s(x, \_)$
- if $\alpha = r(\_, x)$ and $I = \exists s^- \sqsubseteq \exists r^-$, then $\mathrm{ra}(\alpha, I) = s(\_, x)$
- if $\alpha = r(x, y)$ and $I = s \sqsubseteq r$ or $I = s^- \sqsubseteq r^-$, then $\mathrm{ra}(\alpha, I) = s(x, y)$
- if $\alpha = r(x, y)$ and $I = s \sqsubseteq r^-$ or $I = s^- \sqsubseteq r$, then $\mathrm{ra}(\alpha, I) = s(y, x)$

Note: x and y can be variables, individuals, or the special symbol _

REWRITING ALGORITHM PERFECTREF 1/2

Input: TBox $\mathcal{T}$, conjunctive query q_0 (w.l.o.g. assume no $=$-atom with $\exists$-var)
Output: finite set of CQs (which may use special symbol '_')

$PR := \{\tau(q_0)\}$
repeat until $PR' = PR$
  $PR' := PR$
  for each $q \in PR'$ that has not yet been considered **do**
    for each $\alpha \in q$ and $I \in \mathcal{T}$ **do**
      if $\mathrm{ra}(\alpha, I)$ is defined
        $PR := PR \cup \{q[\alpha/\mathrm{ra}(\alpha, I)]\}$
    for each $\alpha, \beta \in q$ **do**
      if α and β unify
        $PR := PR \cup \{\tau(\mathit{merge}(q, \alpha, \beta))\}$
return PR

Functions τ and *merge* described on next slide

Function τ:

- takes as input a query q
- returns the query obtained from q by replacing each existential variable that occurs only once in q by '_'

Atoms α and β unify: exists a substitution ν mapping variables to terms such that $\nu(\alpha) = \nu(\beta)$

Function *merge*:

- input: query q and pair of unifiable atoms $\alpha, \beta \in q$
- returns the query q' obtained from q by:
  - applying the most general unifier of α and β to q
  - adding atom $x = t$ if answer variable x was replaced by term t

Note: *merge* decreases number of concept and role atoms and doesn't add any new terms

EXAMPLE: PERFECTREF ALGORITHM

Let $\mathcal{T} = \{r \sqsubseteq s, A \sqsubseteq \exists s^-, B \sqsubseteq A\}$ and $q_0(y) = \exists x\ s(x, y)$.

EXAMPLE: PERFECTREF ALGORITHM

Let $\mathcal{T} = \{r \sqsubseteq s, A \sqsubseteq \exists s^-, B \sqsubseteq A\}$ and $q_0(y) = \exists x\ s(x, y)$.

Initially, $PR = \{\tau(q_0)\} = \{s(\_, y)\}$.

EXAMPLE: PERFECTREF ALGORITHM

Let $\mathcal{T} = \{r \sqsubseteq s, A \sqsubseteq \exists s^-, B \sqsubseteq A\}$ and $q_0(y) = \exists x\ s(x, y)$.

Initially, $PR = \{\tau(q_0)\} = \{s(\_, y)\}$.

First iteration of PerfectRef adds the following queries:

$q_1 = r(\_, y)$ `apply` $r \sqsubseteq s$ `to only atom of` $\tau(q_0)$

$q_2 = A(y)$ `apply` $A \sqsubseteq \exists s^-$ `to only atom of` $\tau(q_0)$

EXAMPLE: PERFECTREF ALGORITHM

Let $\mathcal{T} = \{r \sqsubseteq s, A \sqsubseteq \exists s^-, B \sqsubseteq A\}$ and $q_0(y) = \exists x\ s(x, y)$.

Initially, $PR = \{\tau(q_0)\} = \{s(\_, y)\}$.

First iteration of PerfectRef adds the following queries:

$q_1 = r(\_, y)$ **apply** $r \sqsubseteq s$ **to only atom of** $\tau(q_0)$

$q_2 = A(y)$ **apply** $A \sqsubseteq \exists s^-$ **to only atom of** $\tau(q_0)$

In second iteration, we add

$q_3 = B(y)$ **apply** $B \sqsubseteq A$ **to only atom of** q_2

Third iteration adds nothing new, so the output is $\{\tau(q_0), q_1, q_2, q_3\}$.

This gives following rewriting: $\exists x\, s(x, y) \vee \exists x\, r(x, y) \vee A(y) \vee B(y)$

EXAMPLE: PERFECTREF ALGORITHM

TBox: $(1)\ \exists \mathit{LectOf} \sqsubseteq \mathit{Prof} \quad (2)\ \mathit{LectOf} \sqsubseteq \mathit{InvWith} \quad (3)\ 100S \sqsubseteq \mathit{IntroC}$

query: $q_0(x, y) = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_2 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_3 = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$

EXAMPLE: PERFECTREF ALGORITHM

TBox: $(1)\ \exists \mathit{LectOf} \sqsubseteq \mathit{Prof} \quad (2)\ \mathit{LectOf} \sqsubseteq \mathit{InvWith} \quad (3)\ 100S \sqsubseteq \mathit{IntroC}$

query: $q_0(x, y) = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_2 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_3 = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$

Second iteration adds:

$$q_4 = \mathit{LectOf}(x, \_) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_5 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$
$$q_6 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge 100S(y)$$

EXAMPLE: PERFECTREF ALGORITHM

TBox: $(1)\ \exists \mathit{LectOf} \sqsubseteq \mathit{Prof} \quad (2)\ \mathit{LectOf} \sqsubseteq \mathit{InvWith} \quad (3)\ 100S \sqsubseteq \mathit{IntroC}$

query: $q_0(x, y) = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_2 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_3 = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$

Second iteration adds:

$$q_4 = \mathit{LectOf}(x, \_) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_5 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$
$$q_6 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge 100S(y)$$

Third iteration gives:

$$q_7 = \mathit{LectOf}(x, \_) \wedge \mathit{LectOf}(x, y) \wedge 100S(y)$$
$$q_8 = \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y) \qquad (\texttt{unifying atoms in } q_4)$$

EXAMPLE: PERFECTREF ALGORITHM

TBox: $(1)\ \exists \mathit{LectOf} \sqsubseteq \mathit{Prof} \quad (2)\ \mathit{LectOf} \sqsubseteq \mathit{InvWith} \quad (3)\ 100S \sqsubseteq \mathit{IntroC}$

query: $q_0(x, y) = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_2 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_3 = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$

Second iteration adds:

$$q_4 = \mathit{LectOf}(x, \_) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_5 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$
$$q_6 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge 100S(y)$$

Third iteration gives:

$q_7 = \mathit{LectOf}(x, \_) \wedge \mathit{LectOf}(x, y) \wedge 100S(y)$

$q_8 = \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$ (`unifying atoms in` q_4)

Fourth and final iteration yields:

$q_9 = \mathit{LectOf}(x, y) \wedge 100S(y)$ (`unifying atoms in` q_7)

EXAMPLE: PERFECTREF ALGORITHM

TBox: $(1)\ \exists \mathit{LectOf} \sqsubseteq \mathit{Prof} \quad (2)\ \mathit{LectOf} \sqsubseteq \mathit{InvWith} \quad (3)\ 100S \sqsubseteq \mathit{IntroC}$

query: $q_0(x, y) = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_2 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_3 = \mathit{Prof}(x) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$

Second iteration adds:

$$q_4 = \mathit{LectOf}(x, \_) \wedge \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$$
$$q_5 = \mathit{LectOf}(x, \_) \wedge \mathit{InvWith}(x, y) \wedge 100S(y)$$
$$q_6 = \mathit{Prof}(x) \wedge \mathit{LectOf}(x, y) \wedge 100S(y)$$

Third iteration gives:

$q_7 = \mathit{LectOf}(x, \_) \wedge \mathit{LectOf}(x, y) \wedge 100S(y)$

$q_8 = \mathit{LectOf}(x, y) \wedge \mathit{IntroC}(y)$ (`unifying atoms in` q_4)

Fourth and final iteration yields:

$q_9 = \mathit{LectOf}(x, y) \wedge 100S(y)$ (`unifying atoms in` q_7)

CORRECTNESS OF REWRITING ALGORITHM

Let *rewrite*$(q, \mathcal{T})$ be the **disjunction of all queries in** PerfectRef$(q, \mathcal{T})$, with each _ symbol replaced by a fresh existential variable.

The following result shows the **correctness of PerfectRef**:

> ***Theorem***. *Let q be a CQ, $(\mathcal{T}, \mathcal{A})$ be a* **satisfiable DL-Lite$_R$ KB**, *$\vec{a}$ be a tuple of individuals from $\mathcal{A}$, and $q^r = rewrite(q, \mathcal{T})$. Then*

$$\vec{a} \in \text{cert}(q, (\mathcal{T}, \mathcal{A})) \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{A}} \models q^r(\vec{a})$$

SATISFIABILITY VIA QUERY REWRITING

Our query rewriting approach only works if the input KB is satisfiable.

- thus: also need a way to test KB satisfiability

SATISFIABILITY VIA QUERY REWRITING

Our query rewriting approach only works if the input KB is satisfiable.

- thus: also need a way to test KB satisfiability

Satisfiability in DL-Lite$_R$ can also be reduced to database querying.

SATISFIABILITY VIA QUERY REWRITING

Our query rewriting approach only works if the input KB is satisfiable.

- thus: also need a way to test KB satisfiability

Satisfiability in DL-Lite$_R$ can also be reduced to database querying.

Given a **negative inclusion** $B \sqsubseteq \neg C$, we denote by $\mathit{unsat}(B \sqsubseteq \neg C)$ the CQ that describes when $B \sqsubseteq \neg C$ is not satisfied. For example:

- $\mathit{unsat}(A \sqsubseteq \neg D) = \exists x\ A(x) \wedge D(x)$
- $\mathit{unsat}(\exists r \sqsubseteq \neg \exists s^{-}) = \exists x, y, z\ r(x, y) \wedge s(z, x)$

SATISFIABILITY VIA QUERY REWRITING

Our query rewriting approach only works if the input KB is satisfiable.

- thus: also need a way to test KB satisfiability

Satisfiability in DL-Lite$_R$ can also be reduced to database querying.

Given a **negative inclusion** $B \sqsubseteq \neg C$, we denote by $\mathit{unsat}(B \sqsubseteq \neg C)$ the CQ that describes when $B \sqsubseteq \neg C$ is not satisfied. For example:

- $\mathit{unsat}(A \sqsubseteq \neg D) = \exists x\ A(x) \wedge D(x)$
- $\mathit{unsat}(\exists r \sqsubseteq \neg \exists s^-) = \exists x, y, z\ r(x, y) \wedge s(z, x)$

Evaluate the following disjunction of Boolean CQs in $\mathcal{I}_\mathcal{A}$:

$$\bigvee_{B \sqsubseteq \neg C \in \mathcal{T}} \text{rewrite}(\mathit{unsat}(B \sqsubseteq \neg C), \mathcal{T})$$

Evaluation returns yes $\Leftrightarrow$ $(\mathcal{T}, \mathcal{A})$ is unsatisfiable

COMPLEXITY OF REASONING IN DL-LITE

Satisfiability and instance checking are tractable:

Theorem. For DL-Lite$_R$, satisfiability and instance checking are NLOGSPACE-complete.

NLOGSPACE $\subseteq$ PTIME

COMPLEXITY OF REASONING IN DL-LITE

Satisfiability and instance checking are tractable:

Theorem. For DL-Lite$_R$, satisfiability and instance checking are NLOGSPACE-complete.

NLOGSPACE $\subseteq$ PTIME

What about ontology-mediated query answering?

Conjunctive query answering is NP-complete already for databases (no TBox). The same is true in DL-Lite:

Theorem. For DL-Lite$_R$, CQ answering is NP-complete.

(note: widely believed NP $\not\subseteq$ PTIME)

DATA COMPLEXITY OF QUERYING IN DL-LITE

NP usually means intractable, yet database queries run fine..

Distinguish two ways of measuring complexity:

- combined complexity: in terms of the size of KB and query
- data complexity: only in terms of the size of the ABox
  - appropriate when $|\mathcal{A}|$ much bigger than $|\mathcal{T}|$, $|q|$ (often the case)

Results stated so far: combined complexity measure

DATA COMPLEXITY OF QUERYING IN DL-LITE

NP usually means intractable, yet database queries run fine..

Distinguish two ways of measuring complexity:

- combined complexity: in terms of the size of KB and query
- data complexity: only in terms of the size of the ABox
  - appropriate when $|\mathcal{A}|$ much bigger than $|\mathcal{T}|$, $|q|$ (often the case)

Results stated so far: combined complexity measure

For the data complexity measure, querying in DL-Lite is tractable:

Theorem. For DL-Lite$_R$, CQ answering is in AC^0 for data complexity.

Note: $AC^0 \subsetneq LOGSPACE \subseteq NLOGSPACE \subseteq PTIME$