

Ontologies & Description Logics

M2 KNOWLEDGE REPRESENTATION

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

REASONING WITH LIGHTWEIGHT DLS

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 (!)

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

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

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:

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

OWL 2 EL profile includes all these extensions

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:

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

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

where:

- C, D, E are arbitrary \mathcal{EL} concepts,
- \hat{D}, \hat{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 $\exists r.(\exists s.A \sqcap H) \sqsubseteq B \sqcap D$

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

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

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

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

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

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

$$\frac{}{A \sqsubseteq A} \text{ T1}$$

$$\frac{}{A \sqsubseteq \top} \text{ T2}$$

$$\frac{A \sqsubseteq B \quad B \sqsubseteq D}{A \sqsubseteq D} \text{ T3}$$

$$\frac{A \sqsubseteq B_1 \quad A \sqsubseteq B_2 \quad B_1 \sqcap B_2 \sqsubseteq D}{A \sqsubseteq D} \text{ T4}$$

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

Rules for deriving assertions

$$\frac{A \sqsubseteq B \quad A(c)}{B(c)} \text{ A1}$$

$$\frac{A_1 \sqcap A_2 \sqsubseteq B \quad A_1(c) \quad A_2(c)}{B(c)} \text{ A2}$$

$$\frac{\exists r.A \sqsubseteq B \quad r(c, d) \quad A(d)}{B(c)} \text{ A3}$$

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 $\alpha_1, \dots, \alpha_n$ and conclusion β is **applicable in \mathcal{K}** if $\{\alpha_1, \dots, \alpha_n\} \subseteq \mathcal{K}$ 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) $\exists \text{hasIngred.Spicy} \sqsubseteq \text{Spicy}$ | (2) $\text{Spicy} \sqcap \text{Dish} \sqsubseteq \text{SpicyDish}$ |
| (3) $\text{ArrabSauce} \sqsubseteq \exists \text{hasIngred.Chili}$ | (4) $\text{Chili} \sqsubseteq \text{Spicy}$ |

ABox \mathcal{A} contains:

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

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 $\text{sat}(\mathcal{K})$ or $\text{sat}(\mathcal{T}, \mathcal{A})$ (resp. $\text{sat}(\mathcal{T})$) 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 $\text{sat}(\mathcal{K})$ or $\text{sat}(\mathcal{T}, \mathcal{A})$ (resp. $\text{sat}(\mathcal{T})$) 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 $\text{sat}(\mathcal{T}', \mathcal{A})$
2. Return **all individuals c such that $A(c) \in \text{sat}(\mathcal{T}', \mathcal{A})$** .

USING SATURATED KB FOR REASONING

Denote by $\text{sat}(\mathcal{K})$ or $\text{sat}(\mathcal{T}, \mathcal{A})$ (resp. $\text{sat}(\mathcal{T})$) 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 $\text{sat}(\mathcal{T}', \mathcal{A})$
2. Return **all individuals c such that $A(c) \in \text{sat}(\mathcal{T}', \mathcal{A})$** .

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

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

USING SATURATED KB FOR REASONING

Denote by $\text{sat}(\mathcal{K})$ or $\text{sat}(\mathcal{T}, \mathcal{A})$ (resp. $\text{sat}(\mathcal{T})$) 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 $\text{sat}(\mathcal{T}', \mathcal{A})$
2. Return **all individuals c such that $A(c) \in \text{sat}(\mathcal{T}', \mathcal{A})$** .

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

1. Normalize \mathcal{T} , yielding \mathcal{T}' , then construct $\text{sat}(\mathcal{T}')$
(can alternatively construct $\text{sat}(\mathcal{T}', \mathcal{A})$ if have an ABox \mathcal{A})
2. Check if $\text{sat}(\mathcal{T}')$ 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 $\text{sat}(\mathcal{K})$ 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 $\text{sat}(\mathcal{K})$ 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** $A \sqsubseteq B \in \text{sat}(\mathcal{K})$ **iff** $A \sqsubseteq B \in \text{sat}(\mathcal{T})$

PROPERTIES OF SATURATION PROCEDURE

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

Theorem. The saturated KB $\text{sat}(\mathcal{K})$ 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 $A \sqsubseteq B \in \text{sat}(\mathcal{K})$ iff $A \sqsubseteq B \in \text{sat}(\mathcal{T})$
- for every ABox assertion $\alpha = A(b)$ with A a concept name:
 $\mathcal{K} \models A(b)$ iff $A(b) \in \text{sat}(\mathcal{K})$

Next slides: sketch proofs for correctness and completeness

CORRECTNESS OF SATURATION

Aim to show that:

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

CORRECTNESS OF SATURATION

Aim to show that:

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

As $\text{sat}(\mathcal{K})$ 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 $\alpha_1, \dots, \alpha_n$, and $\mathcal{K} \models \alpha_i$ ($1 \leq i \leq n$), then $\mathcal{K} \models \beta$.

CORRECTNESS OF SATURATION

Aim to show that:

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

As $\text{sat}(\mathcal{K})$ 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 $\alpha_1, \dots, \alpha_n$, 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 $A \sqsubseteq B \notin \text{sat}(\mathcal{K})$, then $\mathcal{T} \not\models A \sqsubseteq B$
- if $A(b) \notin \text{sat}(\mathcal{K})$, then $\mathcal{K} \not\models A(b)$

We prove the contrapositive, namely:

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

Proof strategy:

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

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

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

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

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

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

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

Observe that by construction, we have:

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

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

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

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

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!

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

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**

We can add all of the following without losing tractability:

- \perp
- $\text{dom}(r) \sqsubseteq C, \text{range}(r) \sqsubseteq C$
- $r_1 \circ \dots \circ r_n \sqsubseteq r_{n+1}$ (complex role inclusions)

EXTENSIONS OF EL

We can add all of the following without losing tractability:

- \perp
- $\text{dom}(r) \sqsubseteq C, \text{range}(r) \sqsubseteq C$
- $r_1 \circ \dots \circ r_n \sqsubseteq r_{n+1}$ (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 ($\text{funct } r$)
- inverse roles r^-

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 too simple – want expressive queries like in DBs
 - **conjunctive queries** ~ **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 too simple – want expressive queries like in DBs
 - **conjunctive queries** ~ **select-project-join** queries in SQL

DL-Lite family: designed for efficient conjunctive query answering

We consider the **dialect DL-Lite_R** (basis for **OWL 2 QL profile**)

- sometimes **abbreviate** to just 'DL-Lite'

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

We consider the **dialect DL-Lite_R** (basis for **OWL 2 QL profile**)

- sometimes **abbreviate** to just 'DL-Lite'

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: $\text{directorOf} \sqsubseteq \text{memberOf}$

We consider the **dialect DL-Lite_R** (basis for **OWL 2 QL profile**)

- sometimes **abbreviate** to just 'DL-Lite'

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: $\text{directorOf} \sqsubseteq \text{memberOf}$

Note: **only basic ABox assertions** ($A(c), r(c, d)$, s.t. A, r concept & role names)

CONJUNCTIVE QUERIES

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

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

CONJUNCTIVE QUERIES

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

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

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

$$q(x_1, \dots, x_k) = \exists y_1, \dots, y_m \alpha_1 \wedge \dots \wedge \alpha_n$$

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

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

Note: where convenient, may treat CQs as sets of atoms, e.g. notation $\alpha \in q$ means α is a conjunct of q

Boolean CQ = CQ that has **no answer variables**

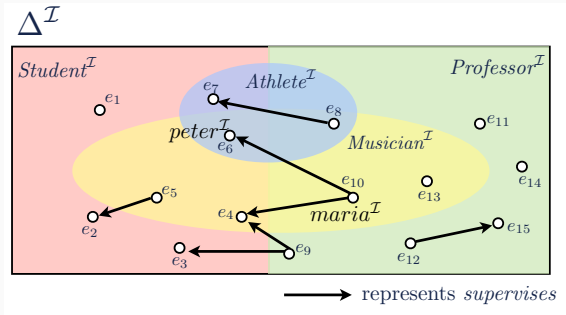
Satisfaction of a Boolean CQ in an interpretation:

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

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

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 .

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:

A tuple $\vec{a} = (a_1, \dots, a_k)$ of individuals from \mathcal{A} is a **certain answer to $q(x_1, \dots, x_k)$ 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 $\text{cert}(q, \mathcal{K})$ the certain answers to q w.r.t. \mathcal{K}

EXAMPLE: CERTAIN ANSWERS

DL-Lite ontology:

$\text{Prof} \sqsubseteq \text{Faculty}$ $\text{Researcher} \sqsubseteq \text{Faculty}$ $\text{Faculty} \sqsubseteq \neg \text{Course}$
 $\text{Prof} \sqsubseteq \exists \text{teaches}$ $\exists \text{teaches}^- \sqsubseteq \text{Course}$

ABox:

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

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

EXAMPLE: CERTAIN ANSWERS

DL-Lite ontology:

$\text{Prof} \sqsubseteq \text{Faculty}$ $\text{Researcher} \sqsubseteq \text{Faculty}$ $\text{Faculty} \sqsubseteq \neg \text{Course}$
 $\text{Prof} \sqsubseteq \exists \text{teaches}$ $\exists \text{teaches}^- \sqsubseteq \text{Course}$

ABox:

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

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

Get the following **certain answers**:

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

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 $q'(\vec{x})$ is an FO-rewriting of a CQ $q(\vec{x})$ w.r.t. \mathcal{T} iff for every ABox \mathcal{A} such that $(\mathcal{T}, \mathcal{A})$ is satisfiable, we have:

$$\vec{a} \in \text{cert}(q, (\mathcal{T}, \mathcal{A})) \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{A}} \models q'(\vec{a})$$

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

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} :

$\text{Prof} \sqsubseteq \text{Faculty}$ $\text{Researcher} \sqsubseteq \text{Faculty}$ $\text{Faculty} \sqsubseteq \neg \text{Course}$
 $\text{Prof} \sqsubseteq \exists \text{teaches}$ $\exists \text{teaches}^- \sqsubseteq \text{Course}$

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

EXAMPLE: QUERY REWRITING IN DL-LITE

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

$\text{Prof} \sqsubseteq \text{Faculty}$ $\text{Researcher} \sqsubseteq \text{Faculty}$ $\text{Faculty} \sqsubseteq \neg \text{Course}$
 $\text{Prof} \sqsubseteq \exists \text{teaches}$ $\exists \text{teaches}^- \sqsubseteq \text{Course}$

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

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

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

EXAMPLE: QUERY REWRITING IN DL-LITE

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

$\text{Prof} \sqsubseteq \text{Faculty}$ $\text{Researcher} \sqsubseteq \text{Faculty}$ $\text{Faculty} \sqsubseteq \neg \text{Course}$
 $\text{Prof} \sqsubseteq \exists \text{teaches}$ $\exists \text{teaches}^- \sqsubseteq \text{Course}$

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

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

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

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

REWRITING ALGORITHM: APPLICABLE AXIOMS

Now we consider how to compute rewritings.

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

REWRITING ALGORITHM: APPLICABLE AXIOMS

Now we consider how to compute rewritings.

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

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^- .

Note: **$_$ is special symbol**, represents **non-shared existential variable**, i.e. which doesn't occur in any other position of the query

REWRITING ALGORITHM: ATOMS

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

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

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

REWRITING ALGORITHM: ATOMS

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

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

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

REWRITING ALGORITHM: ATOMS

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

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

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

REWRITING ALGORITHM: ATOMS

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

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

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

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

Input: TBox \mathcal{T} , conjunctive query q_0

Output: finite set of CQs (possibly with =-atoms and using 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 $l \in \mathcal{T}$ **do**

if $ra(\alpha, l)$ is defined

$PR := PR \cup \{q[\alpha/ra(\alpha, l)]\}$

for each $\alpha, \beta \in q$ **do**

if α and β unify

$PR := PR \cup \{\tau(merge(q, \alpha, \beta))\}$

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

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

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

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

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

PerfectRef first adds the following queries:

$$q_1(y) = r(_, y)$$

apply $r \sqsubseteq s$ to only atom of $\tau(q_0)$

$$q_2(y) = A(y)$$

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

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

PerfectRef first adds the following queries:

$q_1(y) = r(_, y)$	apply $r \sqsubseteq s$ to only atom of $\tau(q_0)$
$q_2(y) = A(y)$	apply $A \sqsubseteq \exists s^-$ to only atom of $\tau(q_0)$

No queries are produced from q_1 , but we get a further query from q_2 :

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

Algorithm returns the set of queries $\{\tau(q_0), q_1, q_2, q_3\}$.

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

(replacing $_$ in $\tau(q_0)$ and q_1 by \exists -var x)

Let $\mathcal{T} = \{A \sqsubseteq \exists r\}$ and $q_0(x, z) = \exists y \, r(x, y) \wedge r(z, y) \wedge B(z)$

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

Let $\mathcal{T} = \{A \sqsubseteq \exists r\}$ and $q_0(x, z) = \exists y \, r(x, y) \wedge r(z, y) \wedge B(z)$

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

First iteration of PerfectRef adds the following query:

$$q_1 = r(x, _) \wedge B(x) \wedge z = x \quad \text{merge operation followed by } \tau$$

Let $\mathcal{T} = \{A \sqsubseteq \exists r\}$ and $q_0(x, z) = \exists y \, r(x, y) \wedge r(z, y) \wedge B(z)$

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

First iteration of PerfectRef adds the following query:

$$q_1 = r(x, _) \wedge B(x) \wedge z = x \quad \text{merge operation followed by } \tau$$

In second iteration, we consider q_1 and add

$$q_2 = A(x) \wedge B(x) \wedge z = x \quad \text{apply } A \sqsubseteq \exists r \text{ to } r\text{-atom of } q_1$$

Output is $\{q_0, q_1, q_2\}$.

This gives the following rewriting: (replacing $_$ in q_1 by \exists -var y)

$$(\exists y \, r(x, y) \wedge r(z, y) \wedge B(z)) \vee (\exists y \, r(x, y) \wedge B(x) \wedge z = x) \vee (A(x) \wedge B(x) \wedge z = x)$$

Equality atoms ensure answer vars (x, z) occur in every disjunct

Consider $\mathcal{T} = \{ \exists \text{LectOf} \sqsubseteq \text{Prof} \quad \text{LectOf} \sqsubseteq \text{InvWith} \quad 100S \sqsubseteq \text{IntroC} \}$
and $q_0(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

Consider $\mathcal{T} = \{ \exists \text{LectOf} \sqsubseteq \text{Prof} \quad \text{LectOf} \sqsubseteq \text{InvWith} \quad 100S \sqsubseteq \text{IntroC} \}$
and $q_0(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1(x, y) = \text{LectOf}(x, _) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$$

$$q_2(x, y) = \text{Prof}(x) \wedge \text{LectOf}(x, y) \wedge \text{IntroC}(y)$$

$$q_3(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge 100S(y)$$

Consider $\mathcal{T} = \{ \exists \text{LectOf} \sqsubseteq \text{Prof} \quad \text{LectOf} \sqsubseteq \text{InvWith} \quad 100S \sqsubseteq \text{IntroC} \}$
and $q_0(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$ (**note:** $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1(x, y) = \text{LectOf}(x, _) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$$

$$q_2(x, y) = \text{Prof}(x) \wedge \text{LectOf}(x, y) \wedge \text{IntroC}(y)$$

$$q_3(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge 100S(y)$$

From preceding queries, we get:

$$q_4(x, y) = \text{LectOf}(x, _) \wedge \text{LectOf}(x, y) \wedge \text{IntroC}(y)$$

$$q_5(x, y) = \text{LectOf}(x, _) \wedge \text{InvWith}(x, y) \wedge 100S(y)$$

$$q_6(x, y) = \text{Prof}(x) \wedge \text{LectOf}(x, y) \wedge 100S(y)$$

Consider $\mathcal{T} = \{ \exists \text{LectOf} \sqsubseteq \text{Prof} \quad \text{LectOf} \sqsubseteq \text{InvWith} \quad 100S \sqsubseteq \text{IntroC} \}$
 and $q_0(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1(x, y) = \text{LectOf}(x, _) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$$

$$q_2(x, y) = \text{Prof}(x) \wedge \text{LectOf}(x, y) \wedge \text{IntroC}(y)$$

$$q_3(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge 100S(y)$$

From preceding queries, we get:

$$q_4(x, y) = \text{LectOf}(x, _) \wedge \text{LectOf}(x, y) \wedge \text{IntroC}(y)$$

$$q_5(x, y) = \text{LectOf}(x, _) \wedge \text{InvWith}(x, y) \wedge 100S(y)$$

$$q_6(x, y) = \text{Prof}(x) \wedge \text{LectOf}(x, y) \wedge 100S(y)$$

Further queries obtained when considering q_4 :

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

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

Consider $\mathcal{T} = \{ \exists \text{LectOf} \sqsubseteq \text{Prof} \quad \text{LectOf} \sqsubseteq \text{InvWith} \quad 100S \sqsubseteq \text{IntroC} \}$
 and $q_0(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$ (note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$$q_1(x, y) = \text{LectOf}(x, _) \wedge \text{InvWith}(x, y) \wedge \text{IntroC}(y)$$

$$q_2(x, y) = \text{Prof}(x) \wedge \text{LectOf}(x, y) \wedge \text{IntroC}(y)$$

$$q_3(x, y) = \text{Prof}(x) \wedge \text{InvWith}(x, y) \wedge 100S(y)$$

From preceding queries, we get:

$$q_4(x, y) = \text{LectOf}(x, _) \wedge \text{LectOf}(x, y) \wedge \text{IntroC}(y)$$

$$q_5(x, y) = \text{LectOf}(x, _) \wedge \text{InvWith}(x, y) \wedge 100S(y)$$

$$q_6(x, y) = \text{Prof}(x) \wedge \text{LectOf}(x, y) \wedge 100S(y)$$

Further queries obtained when considering q_4 :

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

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

Final iteration yields:

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

PROPERTIES OF REWRITING ALGORITHM

Lemma The algorithm PerfectRef **always terminates**.

Proof idea: Can **bound number of queries produced**, as generated queries have at most as many concept and role atoms as input query and only use symbols from query or TBox (or special symbol '_').

PROPERTIES OF REWRITING ALGORITHM

Lemma The algorithm PerfectRef **always terminates**.

Proof idea: Can **bound number of queries produced**, as generated queries have at most as many concept and role atoms as input query and only use symbols from query or TBox (or special symbol '_').

Let $\text{rewrite}(q, \mathcal{T})$ 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 $q(\vec{x})$ be a CQ (without \exists -vars in equality atoms), $(\mathcal{T}, \mathcal{A})$ be a satisfiable DL-Lite_R KB, \vec{a} be a tuple of individuals from \mathcal{A} with $|\vec{x}| = |\vec{a}|$, and $q^r = \text{rewrite}(q, \mathcal{T})$. 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 $\text{unsat}(B \sqsubseteq \neg C)$ the CQ that describes when $B \sqsubseteq \neg C$ is not satisfied. For example:

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

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 $\text{unsat}(B \sqsubseteq \neg C)$ the CQ that describes when $B \sqsubseteq \neg C$ is not satisfied. For example:

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

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

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

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

Satisfiability and instance checking are tractable:

Theorem. For DL-Lite_R , satisfiability and instance checking are **NLOGSPACE-complete**.

$$\text{NLOGSPACE} \subseteq \text{PTIME}$$

Satisfiability and instance checking are tractable:

Theorem. For DL-Lite_R , satisfiability and instance checking are **NLOGSPACE-complete**.

$$\text{NLOGSPACE} \subseteq \text{PTIME}$$

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 $\text{NP} \not\subseteq \text{PTIME}$)

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

Follows from AC^0 data complexity of FO-query evaluation

CLOSER LOOK AT QUERY REWRITING & CQ ANSWERING

Many of the proposed rewriting algorithms produce **unions of conjunctive queries** (UCQs = \vee of CQs)

SIZE OF REWRITINGS

Many of the proposed rewriting algorithms produce **unions of conjunctive queries** (UCQs = \vee of CQs)

Not hard to see **smallest UCQ-rewriting may be exponentially large:**

Many of the proposed rewriting algorithms produce **unions of conjunctive queries (UCQs = \vee of CQs)**

Not hard to see **smallest UCQ-rewriting may be exponentially large:**

- Query: $A_1^0(x) \wedge \dots \wedge A_n^0(x)$
- Ontology: $A_1^1(x) \rightarrow A_1^0(x) \quad A_2^1(x) \rightarrow A_2^0(x) \quad \dots \quad A_n^1(x) \rightarrow A_n^0(x)$
- Rewriting: $\bigvee_{(i_1, \dots, i_n) \in \{0,1\}^n} A_1^{i_1}(x) \wedge A_2^{i_2}(x) \wedge \dots \wedge A_n^{i_n}(x)$

SIZE OF REWRITINGS

Many of the proposed rewriting algorithms produce **unions of conjunctive queries (UCQs = \vee of CQs)**

Not hard to see **smallest UCQ-rewriting may be exponentially large:**

- Query: $A_1^0(x) \wedge \dots \wedge A_n^0(x)$
- Ontology: $A_1^1(x) \rightarrow A_1^0(x) \quad A_2^1(x) \rightarrow A_2^0(x) \quad \dots \quad A_n^1(x) \rightarrow A_n^0(x)$
- Rewriting: $\bigvee_{(i_1, \dots, i_n) \in \{0,1\}^n} A_1^{i_1}(x) \wedge A_2^{i_2}(x) \wedge \dots \wedge A_n^{i_n}(x)$

But: **simple polysize FO-rewriting does exist!** $\bigwedge_{i=1}^n (A_i^0(x) \vee A_i^1(x))$

SIZE OF REWRITINGS

Many of the proposed rewriting algorithms produce **unions of conjunctive queries** (UCQs = \vee of CQs)

Not hard to see **smallest UCQ-rewriting may be exponentially large**:

- Query: $A_1^0(x) \wedge \dots \wedge A_n^0(x)$
- Ontology: $A_1^1(x) \rightarrow A_1^0(x) \quad A_2^1(x) \rightarrow A_2^0(x) \quad \dots \quad A_n^1(x) \rightarrow A_n^0(x)$
- Rewriting: $\bigvee_{(i_1, \dots, i_n) \in \{0,1\}^n} A_1^{i_1}(x) \wedge A_2^{i_2}(x) \wedge \dots \wedge A_n^{i_n}(x)$

But: **simple polysize FO-rewriting does exist!** $\bigwedge_{i=1}^n (A_i^0(x) \vee A_i^1(x))$

To get positive results, **need to go beyond UCQs**

DIFFERENT FORMS OF REWRITINGS

PE-rewritings: **positive existential queries** (only \exists , \wedge , \vee)

$$(r(x, y) \vee s(y, x)) \wedge (A(x) \vee (B(x) \wedge \exists z p(x, z))) \wedge (A(y) \vee (B(y) \wedge \exists z p(y, z)))$$

DIFFERENT FORMS OF REWRITINGS

PE-rewritings: **positive existential queries** (only \exists , \wedge , \vee)

$$(r(x, y) \vee s(y, x)) \wedge (A(x) \vee (B(x) \wedge \exists z p(x, z))) \wedge (A(y) \vee (B(y) \wedge \exists z p(y, z)))$$

NDL-rewritings: **non-recursive Datalog queries**

$$q_1(x, y), q_2(x), q_2(y) \rightarrow \text{goal}(x, y)$$

$$r(x, y) \rightarrow q_1(x, y)$$

$$s(y, x) \rightarrow q_1(x, y)$$

$$A(x) \rightarrow q_2(x)$$

$$B(x), p(x, z) \rightarrow q_2(x)$$

DIFFERENT FORMS OF REWRITINGS

PE-rewritings: **positive existential queries** (only \exists , \wedge , \vee)

$$(r(x, y) \vee s(y, x)) \wedge (A(x) \vee (B(x) \wedge \exists z p(x, z))) \wedge (A(y) \vee (B(y) \wedge \exists z p(y, z)))$$

NDL-rewritings: **non-recursive Datalog queries**

$$q_1(x, y), q_2(x), q_2(y) \rightarrow \text{goal}(x, y)$$

$$r(x, y) \rightarrow q_1(x, y)$$

$$A(x) \rightarrow q_2(x)$$

$$s(y, x) \rightarrow q_1(x, y)$$

$$B(x), p(x, z) \rightarrow q_2(x)$$

FO-rewritings: **first-order queries** (can also use \forall , \neg)

DIFFERENT FORMS OF REWRITINGS

PE-rewritings: **positive existential queries** (only \exists , \wedge , \vee)

$$(r(x, y) \vee s(y, x)) \wedge (A(x) \vee (B(x) \wedge \exists z p(x, z))) \wedge (A(y) \vee (B(y) \wedge \exists z p(y, z)))$$

NDL-rewritings: **non-recursive Datalog queries**

$$q_1(x, y), q_2(x), q_2(y) \rightarrow \text{goal}(x, y)$$

$$r(x, y) \rightarrow q_1(x, y)$$

$$A(x) \rightarrow q_2(x)$$

$$s(y, x) \rightarrow q_1(x, y)$$

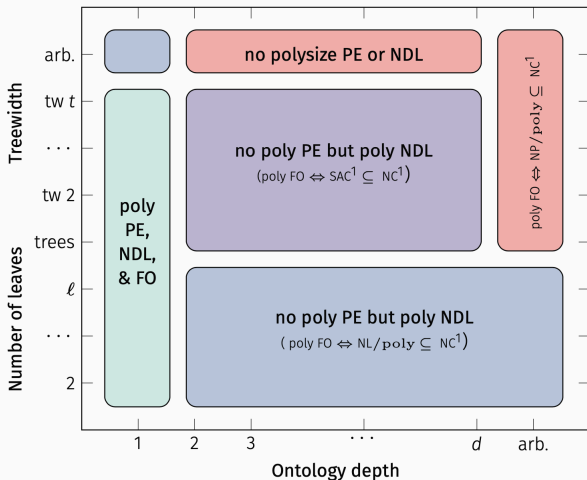
$$B(x), p(x, z) \rightarrow q_2(x)$$

FO-rewritings: **first-order queries** (can also use \forall , \neg)

What if we replace UCQs by PE / NDL / FO?

Do we get polysize rewritings?

GLIMPSE AT SUCCINCTNESS LANDSCAPE



- cannot guarantee polysize rewritings in general case
- but polysize NDL-rewritings do exist in relevant settings

Adopt **NDL-rewritings** to avoid combinatorial explosion

Optimizations to further reduce rewriting size

- **exploit structure of data (e.g. satisfied constraints)** which make some parts of rewriting superfluous

Pre-computation when possible

- add all inferred ABox assertions
- **combined approach**: store **compact canonical model**, then filter answers to remove false positives

Example system: **Ontop** (ontop-vkg.org)

EXTENDING THE REACH OF FO-REWRITING

For many ontology languages, **FO-rewritings not guaranteed to exist**

- consider query $A(x)$ and \mathcal{EL} -TBox $\{\exists r.A \sqsubseteq A\}$

But this is **worst-case result**, sometimes FO-rewriting is possible!

EXTENDING THE REACH OF FO-REWRITING

For many ontology languages, **FO-rewritings not guaranteed to exist**

- consider query $A(x)$ and \mathcal{EL} -TBox $\{\exists r.A \sqsubseteq A\}$

But this is **worst-case result**, sometimes FO-rewriting is possible!

To **extend the applicability of FO-query rewriting** beyond DL-Lite:

- devise **methods of identifying ‘good cases’** (static analysis task)
 - given \mathcal{O} and q , does q have an FO-rewriting w.r.t. \mathcal{O} ?
- **construct rewritings** when they exist

EXTENDING THE REACH OF FO-REWRITING

For many ontology languages, **FO-rewritings not guaranteed to exist**

- consider query $A(x)$ and \mathcal{EL} -TBox $\{\exists r.A \sqsubseteq A\}$

But this is **worst-case result**, sometimes FO-rewriting is possible!

To **extend the applicability of FO-query rewriting** beyond DL-Lite:

- devise **methods of identifying ‘good cases’** (static analysis task)
 - given \mathcal{O} and q , does q have an FO-rewriting w.r.t. \mathcal{O} ?
- **construct rewritings** when they exist

Good news: **FO-rewritability decidable for many DLs**

- between EXPTIME and 2NEXPTIME-complete, depending on the DL

Moreover: **practical algorithm for \mathcal{EL}**

- FO-rewritings **often exist** and can be **efficiently computed**