Ontologies and Description Logics

Parcours IA - Représentation des connaissances

Meghyn Bienvenu, CNRS researcher at LaBRI
Reasoning Techniques for Lightweight DLs
1. EL
The $\mathcal{EL}$ family

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 (!)

More about SNOMED: http://www.ihtsdo.org/snomed-ct/
Syntax of $\mathcal{EL}$

The basic logic $\mathcal{EL}$ allows complex concepts of the following form:

$$C := \top \mid A \mid C_1 \cap C_2 \mid \exists R.C$$

Inclusions $C_1 \sqsubseteq C_2$ and assertions $A(c), R(c, d)$

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: $R \circ R \sqsubseteq R$)

$\text{OWL 2 EL}$ includes all these extensions.

Today we will focus on plain $\mathcal{EL}$ (without these extensions).
Normal form for $\mathcal{EL}$ TBoxes

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

\[ A_1 \cap A_2 \sqsubseteq B \quad A \sqsubseteq \exists R.B \quad \exists R.A \sqsubseteq B \]

where $A, A_1, A_2, B$ are atomic concepts (or $\top$).
Normal form for $\mathcal{EL}$ TBoxes

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

$$A_1 \cap A_2 \sqsubseteq B \quad A \sqsubseteq \exists R.B \quad \exists R.A \sqsubseteq B$$

where $A, A_1, A_2, B$ are atomic concepts (or $\top$).

**Fact:** for every TBox $\mathcal{EL}$ $\mathcal{T}$, we can construct in polynomial time a TBox $\mathcal{T}'$ in normal form (possibly using extra atomic concepts) such that:

- for every inclusion $C \sqsubseteq D$ which uses only atomic concepts 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 $\alpha$ that only uses atomic concepts from $(\mathcal{T}, \mathcal{A})$, we have $\mathcal{T}, \mathcal{A} \models \alpha$ iff $\mathcal{T}', \mathcal{A} \models \alpha$
Normalization procedure

Apply the following rules exhaustively to $\mathcal{T}$, until none is applicable:

- replace $C \equiv D$ by $C \sqsubseteq D$ and $D \sqsubseteq C$
- replace $C \sqsubseteq D_1 \cap D_2$ by $C \sqsubseteq D_1$ and $C \sqsubseteq D_2$
- if $\exists R.C$ occurs in $\mathcal{T}$ and $C$ is complex, replace all occurrences of $C$ by fresh atomic concept $X_C$, and add $X_C \sqsubseteq C$ and $C \sqsubseteq X_C$
- if $C \sqsubseteq D \in \mathcal{T}$, $C$ contains $\exists R.B$, and $C \neq \exists R.B$, then replace $C \sqsubseteq D$ by $\{\exists R.B \sqsubseteq X, C^X \sqsubseteq D\}$, where $C^X$ is $C$ with $\exists R.B$ replaced by $X$.
- if $A_1 \cap \ldots \cap A_n \sqsubseteq C \in \mathcal{T}$ with $n > 2$, then replace this inclusion by $\{A_2 \cap \ldots \cap A_n \sqsubseteq X, A_1 \cap X \sqsubseteq C\}$, with $X$ a fresh atomic concept.
- if $\exists R.C \sqsubseteq \exists S.D \in \mathcal{T}$, then replace this inclusion by $\{\exists R.C \sqsubseteq X, X \sqsubseteq \exists S.D\}$, with $X$ a fresh atomic concept.
Saturation rules for EL

Deriving new ontology axioms

\[
\begin{align*}
A & \sqsubseteq B_i \ (1 \leq i \leq n) & B_1 \sqcap \ldots \sqcap B_n & \sqsubseteq D \\
A & \sqsubseteq D & A & \sqsubseteq D \\
A & \sqsubseteq \exists R.B & B & \sqsubseteq D & \exists R.D & \sqsubseteq E & A & \sqsubseteq E
\end{align*}
\]

01 \hspace{1cm} 02

03

Deriving new facts

\[
\begin{align*}
A_1 \sqcap \ldots \sqcap A_n & \sqsubseteq B \\
A_i(c) \ (1 \leq i \leq n) & \hspace{1cm} \exists R.B & \sqsubseteq A \\
B(c) & \hspace{1cm} R(c, d) & \sqsubseteq B(d) \\
B(c) & \hspace{1cm} A(c)
\end{align*}
\]

D1 \hspace{1cm} D2

Apply rules exhaustively, denote resulting KB by \( sat(\mathcal{K}) \) or \( sat(\mathcal{T}, \mathcal{A}) \)
Using \( \text{sat}(\mathcal{K}) \) for reasoning

To find all instances of atomic concept \( A \) 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 \( sat(\mathcal{K}) \) for reasoning

To find all instances of atomic concept \( A \) 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 decide subsumption between atomic concepts for a TBox \( \mathcal{T} \):
1. Normalize \( \mathcal{T} \), yielding \( \mathcal{T}' \), then construct \( sat(\mathcal{T}', \emptyset) \)
2. To test whether \( \mathcal{T} \models A \sqsubseteq B \), check if \( sat(\mathcal{T}', \emptyset) \) contains \( A \sqsubseteq B \), return yes if so, else no.

▶ use new atomic concepts to represent complex concepts, e.g. if \( C \) is a complex concept, add \( X_C \subseteq C \) and \( C \subseteq X_C \) to \( \mathcal{T} \) (with \( X_C \) a fresh atomic concept).

▶ proceed as above, checking for \( X_C \sqsubseteq X_D \) in place of \( C \sqsubseteq D \).
Using \( \text{sat}(K) \) for reasoning

To find all instances of atomic concept \( A \) w.r.t. \( K = (T, A) \):

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

To decide subsumption between atomic concepts for a TBox \( T \):

1. Normalize \( T \), yielding \( T' \), then construct \( \text{sat}(T', \emptyset) \)
2. To test whether \( T \models A \sqsubseteq B \), check if \( \text{sat}(T', \emptyset) \) contains \( A \sqsubseteq B \), return yes if so, else no.

To decide subsumption between general concepts:

- use new atomic concepts to represent complex concepts, e.g. if \( C \) is a complex concept, add \( X_C \subseteq C \) and \( C \subseteq X_C \) to \( T \) (with \( X_C \) a fresh atomic concept).
- proceed as above, checking for \( X_C \sqsubseteq X_D \) in place of \( C \sqsubseteq D \).
Example: Saturation in EL

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 O3
9. \( \text{Spicy}(s) \) using (7), (8), and rule D1
10. \( \text{Spicy}(p) \) using (1), (6), (9), and rule D2
11. \( \text{SpicyDish}(p) \) using (2), (5), (10), and rule D1

Examining the result, will return \( p \) as answer to \( q(x) = \text{SpicyDish}(x) \).
Example: Saturation in EL

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 O3
(9) $\text{Spicy}(s)$ using (7), (8), and rule D1
(10) $\text{Spicy}(p)$ using (1), (6), (9), and rule D2
(11) $\text{SpicyDish}(p)$ using (2), (5), (10), and rule D1

Examining the result, will return $p$ as answer to $q(x) = \text{SpicyDish}(x)$.
Example: Saturation in EL

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. $\text{ArrabSauce}(s)$

Saturation procedure adds the following axioms and assertions:

8. $\text{ArrabSauce} \sqsubseteq \text{Spicy}$ using (1), (3), (4) and rule O3
9. $\text{Spicy}(s)$ using (7), (8), and rule D1
10. $\text{Spicy}(p)$ using (1), (6), (9), and rule D2
11. $\text{SpicyDish}(p)$ using (2), (5), (10), and rule D1

Examining the result, will return $p$ as answer to $q(x) = \text{SpicyDish}(x)$
Exercise: Saturation in EL

Consider the following KB $\mathcal{K}$ (whose TBox is in normal form):

$\mathcal{T} = \{ A \sqcap D \sqsubseteq E, D \sqsubseteq F, B \sqsubseteq \exists S.A, B \sqsubseteq \exists S.D, \exists S.E \sqsubseteq G, \exists S.H \sqsubseteq D \}$

$\mathcal{A} = \{ S(u, v), S(v, u), A(v), H(u) \}$

For each of the following, determine whether it belongs to $\text{sat}(\mathcal{K})$

(1) $B \sqsubseteq \exists S.(A \sqcap D)$  (2) $J \sqsubseteq \exists S.A$
(3) $B \sqsubseteq D$  (4) $\exists S.(A \sqcap D) \sqsubseteq G$
(5) $B \sqsubseteq G$  (6) $D(u)$
(7) $E(u)$  (8) $H(u)$
(9) $A(v)$  (10) $G(v)$
Complexity of reasoning in $\mathcal{EL}$

**Theorem.** Subsumption and instance checking in $\mathcal{EL}$ are PTIME-complete.

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

**Note:** with only $\sqcap$ and $\forall R.C$, these problems are EXPTIME-complete!
Complexity of reasoning in $\mathcal{EL}$

**Theorem.** Subsumption and instance checking in $\mathcal{EL}$ are PTIME-complete.

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

**Note:** with only $\sqcap$ and $\forall R.C$, these problems are EXPTIME-complete!

In practice:
- huge ontologies like SNOMED can be classified in a few seconds
Extensions of $\mathcal{EL}$

We can add all of the following without losing tractability:

- $\bot$
- $\text{dom}(R) \subseteq C$, $\text{range}(R) \subseteq C$
- $R_1 \circ \ldots \circ R_n \subseteq R_{n+1}$
Extensions of $\mathcal{EL}$

We can add all of the following without losing tractability:

- $\bot$
- $\text{dom}(R) \subseteq C$, $\text{range}(R) \subseteq C$
- $R_1 \circ \ldots \circ R_n \subseteq R_{n+1}$

However, just one of the following makes subsumption EXPTIME-hard:

- negation $\neg$
- disjunction $\sqcup$
- at-least or at-most restrictions: $\geq 2R$, $\leq 1R$
- functional roles ($\text{funct} R$)
- inverse roles $R^-$
2. *DL-Lite*
Using ontologies to access data

**Objective:** enrich standard relational databases (DBs) with ontologies

- ontology provides a *convenient vocabulary* for users to specify queries
- ontology can be used to link *multiple datasets with different schemas*
- knowledge in ontology can yield *additional answers to queries*
Using ontologies to access data

**Objective:** enrich standard relational databases (DBs) with ontologies

▶ ontology provides a *convenient vocabulary* for users to specify queries
▶ ontology can be used to 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*
Using ontologies to access data

**Objective**: enrich standard relational databases (DBs) with ontologies

- ontology provides a convenient vocabulary for users to specify queries
- ontology can be used to 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

**Objective:** enrich standard relational databases (DBs) with ontologies

- ontology provides a convenient vocabulary for users to specify queries
- ontology can be used to 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
Syntax of \textbf{DL-Lite}

We present the dialect $DL-Lite_{\mathcal{R}}$ (which underlies OWL2 QL).

\textbf{ABox assertions}: $A(c)$, $R(c, d)$

\textbf{TBox inclusions}: $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$, $S_1 \sqsubseteq S_2$, $S_1 \sqsubseteq \neg S_2$ where

$$B := A \mid \exists S \quad S := R \mid R^-$$

with $A$ an atomic concept and $R$ an atomic role
We present the dialect $DL-Lite_R$ (which underlies OWL2 QL).

**ABox assertions:** $A(c), R(c, d)$

**TBox inclusions:** $B_1 \sqsubseteq B_2, B_1 \sqsubseteq \neg B_2, S_1 \sqsubseteq S_2, S_1 \sqsubseteq \neg S_2$ where

$$B := A \mid \exists S \quad S := R \mid R^-$$

with $A$ an atomic concept and $R$ an atomic role

Other $DL-Lite$ dialects allow:

- functional roles (funct $S$)
- cardinality restrictions ($\geq q S, \leq q S$)
- Horn inclusions ($B_1 \sqcap \ldots \sqcap B_n \sqsubseteq (\neg)B_{n+1}$)
- roles which are symmetric, asymmetric, reflexive, or anti-reflexive
Example: Describing families in \textit{DL-Lite}_R

- **subclasses**
  \[ \text{Mother} \sqsubseteq \text{Parent} \quad \text{Father} \sqsubseteq \text{Parent} \quad \text{Mother} \sqsubseteq \text{Female} \]

- **subrelations**
  \[ \text{MotherOf} \sqsubseteq \text{ParentOf} \quad \text{FatherOf} \sqsubseteq \text{ParentOf} \]

- **domain and range constraints**
  \[ \exists \text{ParentOf} \sqsubseteq \text{Parent} \quad \exists \text{ChildOf}\neg \sqsubseteq \text{Parent} \]

- **participation constraints**
  \[ \text{Parent} \sqsubseteq \exists \text{ChildOf}\neg \quad \text{Mother} \sqsubseteq \exists \text{MotherOf} \]

- **inverses of binary relations**
  \[ \text{ParentOf} \sqsubseteq \text{ChildOf}\neg \quad \text{ChildOf} \sqsubseteq \text{ParentOf}\neg \]

- **disjointness**
  \[ \text{Male} \sqsubseteq \neg \text{Female} \quad \text{Childless} \sqsubseteq \neg \exists \text{ParentOf} \]
Conjunctive queries

An atom takes the form $P(t_1, \ldots, t_n)$ where:
- $P$ is a relation symbol
- each $t_i$ is a term ($=$ variable or individual name)

A conjunctive query (CQ) has the form

$$\exists x_1, \ldots, x_m \quad \alpha_1 \land \ldots \land \alpha_r$$

where each $\alpha_i$ is an atom, and each $x_i$ appears in some atom.
- $x_1, \ldots, x_m$ are called quantified variables
- all other variables are called answer variables.

Since we consider DLs, relation symbols are atomic concept and roles.
Example: conjunctive queries

Find all pairs of teachers such that the first is the parent of the second:

$$Teacher(x) \land ParentOf(x, y) \land Teacher(y)$$

Find all teachers that have a child who is a teacher:

$$\exists y \ Teacher(x) \land ParentOf(x, y) \land Teacher(y)$$

Find all teachers that have a parent who is a teacher:

$$\exists x \ Teacher(x) \land ParentOf(x, y) \land Teacher(y)$$

Does there exist a teacher with a child who is a teacher?

$$\exists x, y \ Teacher(x) \land ParentOf(x, y) \land Teacher(y)$$

Does Marie have a child who is a teacher?

$$\exists y \ ParentOf(marie, y) \land Teacher(y)$$
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 $\pi$ 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 $\alpha = P(t_1, \ldots, t_k)$ of $q$,

$$ (\pi(t_1), \ldots, \pi(t_k)) \in P^\mathcal{I} $$
Example: Satisfaction in an interpretation

Reconsider the interpretation $\mathcal{I}$:

$\Delta^{\mathcal{I}} = \text{Person}^{\mathcal{I}}$

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

(1) $\text{ParentOf}(\text{peter, maria})$

(2) $\exists x \text{ParentOf}(x, \text{peter}) \land \text{Female}(x)$

(3) $\exists x, y \text{Male}(x) \land \text{ParentOf}(x, y) \land \text{Female}(y)$

(4) $\exists x \text{Teacher}(x) \land \text{Female}(x)$

(5) $\exists x \text{ParentOf}(x, x)$

(6) $\exists x, y \text{Teacher}(x) \land \text{ParentOf}(x, y) \land \text{Teacher}(y)$
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 $(a_1, \ldots, a_k)$ of individuals from $\mathcal{A}$ is a (certain) answer to $q$ w.r.t. $\mathcal{K}$ if and only if

$$\mathcal{K} \models q[a_1, \ldots, a_k]$$

where $q[a_1, \ldots, a_k]$ is $q$ 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

Consider the TBox:

\[ \text{Parent} \sqsubseteq \exists \text{ParentOf} \quad \exists \text{ParentOf} \sqsubseteq \text{Parent} \quad \text{Mother} \sqsubseteq \text{Parent} \quad \text{Father} \sqsubseteq \text{Parent} \\
\text{Mother} \sqsubseteq \text{Female} \quad \text{Father} \sqsubseteq \text{Male} \quad \text{ParentOf} \sqsubseteq \text{ChildOf}^- \quad \text{ChildOf}^- \sqsubseteq \text{ParentOf} \]

and the ABox:

\[ \text{Mother(}mary\text{)} \quad \text{Father(}paul\text{)} \quad \text{ParentOf(}julie, marc\text{)} \quad \text{ParentOf(}marc, paul\text{)} \]

Determine the certain answers to the following queries.

1. \( \text{Female}(x) \)
2. \( \text{ChildOf}(x, y) \)
3. \( \exists y \text{ ParentOf}(x, y) \)
4. \( \text{ParentOf}(x, y) \land \text{ChildOf}(y, z) \)
5. \( \text{ParentOf}(x, y) \land \text{ChildOf}(z, y) \)
Idea: exploit the efficiency of database (DB) management systems

Data storage: two possibilities

- ABox is stored directly in a traditional database / RDF store ..., or
- ABox is generated from existing DB(s) via mappings (virtual ABox)
  - $\exists y, z, w \text{Employee}(x, y, z, "\text{teacher"}, w) \rightarrow \text{Teacher}(x)$
Query rewriting approach

**Idea:** exploit the efficiency of database (DB) management systems

**Data storage:** two possibilities

- ABox is stored directly in a traditional database / RDF store ..., or
- ABox is generated from existing DB(s) via mappings (virtual ABox)
  - $\exists y, z, w \text{Employee}(x, y, z, "\text{teacher"}, w) \rightarrow \text{Teacher}(x)$

**Query rewriting**

- input query is rewritten into a new query that integrates all relevant information from the TBox (and mappings)
Query rewriting approach

**Idea:** exploit the efficiency of database (DB) management systems

**Data storage:** two possibilities

- ABox is stored directly in a traditional database / RDF store ..., or
- ABox is generated from existing DB(s) via mappings (virtual ABox)
  - $\exists y, z, w \text{Employee}(x, y, z, \text{"teacher"}, w) \rightarrow \text{Teacher}(x)$

**Query rewriting**

- input query is rewritten into a new query that integrates all relevant information from the TBox (and mappings)

**Query evaluation**

- the rewritten query is expressed in a standard DB query language (e.g. SQL) and evaluated over the database(s)
**Query rewriting approach**

**Idea:** exploit the efficiency of database (DB) management systems

**Data storage:** two possibilities

- ABox is stored directly in a traditional database / RDF store ..., or
- ABox is generated from existing DB(s) via mappings (virtual ABox)
  - \( \exists y, z, w \text{Employee}(x, y, z, \text{“teacher”}, w) \rightarrow \text{Teacher}(x) \)

**Query rewriting**

- input query is rewritten into a new query that integrates all relevant information from the TBox (and mappings)

**Query evaluation**

- the rewritten query is expressed in a standard DB query language (e.g. SQL) and evaluated over the database(s)
Query rewriting

Given an ABox $\mathcal{A}$, define the interpretation $\mathcal{I}_\mathcal{A}$ as follows:

- $\Delta^{\mathcal{I}_\mathcal{A}}$ contains the individuals in $\mathcal{A}$
- $A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\}$
- $R^{\mathcal{I}_\mathcal{A}} = \{(a, b) \mid R(a, b) \in \mathcal{A}\}$
- $a^{\mathcal{I}} = a$

Think of $\mathcal{I}_\mathcal{A}$ as the DB representation of $\mathcal{A}$. 
Given an ABox $\mathcal{A}$, define the interpretation $\mathcal{I}_\mathcal{A}$ as follows:

- $\Delta^{\mathcal{I}_\mathcal{A}}$ contains the individuals in $\mathcal{A}$
- $A^{\mathcal{I}_\mathcal{A}} = \{ a \mid A(a) \in \mathcal{A} \}$
- $R^{\mathcal{I}_\mathcal{A}} = \{ (a, b) \mid R(a, b) \in \mathcal{A} \}$
- $a^{\mathcal{I}} = a$

Think of $\mathcal{I}_\mathcal{A}$ as the DB representation of $\mathcal{A}$.

A query $q'$ is called a **rewriting** of a CQ $q$ w.r.t. a TBox $\mathcal{T}$ if and only if

$$\bar{a} \in \text{cert}(q, (\mathcal{T}, \mathcal{A})) \iff \mathcal{I}_\mathcal{A} \models q'[\bar{a}]$$

for every ABox $\mathcal{A}$. 
Query rewriting

Given an ABox $\mathcal{A}$, define the interpretation $\mathcal{I}_\mathcal{A}$ as follows:

- $\Delta^{\mathcal{I}_\mathcal{A}}$ contains the individuals in $\mathcal{A}$
- $A^{\mathcal{I}_\mathcal{A}} = \{ a \mid A(a) \in \mathcal{A} \}$
- $R^{\mathcal{I}_\mathcal{A}} = \{ (a, b) \mid R(a, b) \in \mathcal{A} \}$
- $a^{\mathcal{I}} = a$

Think of $\mathcal{I}_\mathcal{A}$ as the DB representation of $\mathcal{A}$.

A query $q'$ is called a **rewriting** of a CQ $q$ w.r.t. a TBox $\mathcal{T}$ if and only if

$$\bar{a} \in \text{cert}(q, (\mathcal{T}, \mathcal{A})) \iff \mathcal{I}_\mathcal{A} \models q'[\bar{a}]$$

for every ABox $\mathcal{A}$.

**Note:** to exploit DB systems, $q'$ should be a first-order ($\sim$ SQL) query
Example: query rewriting

TBox $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor}$  $\exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith}$  $\text{Assists} \sqsubseteq \text{InvolvedWith}$

$\exists \text{InvolvedWith}^{-} \sqsubseteq \text{Course}$  $\text{IntroCourse} \sqsubseteq \text{Course}$  $\text{100Series} \sqsubseteq \text{IntroCourse}$
Example: query rewriting

TBox $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor}$  
$\exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith}$  
$\exists \text{Assists} \sqsubseteq \text{InvolvedWith}$  
$\exists \text{InvolvedWith}^- \sqsubseteq \text{Course}$  
$\text{IntroCourse} \sqsubseteq \text{Course}$  
$100\text{Series} \sqsubseteq \text{IntroCourse}$

Query $q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$
Example: query rewriting

**TBox** $\mathcal{T}$:

$\exists$LecturerOf $\sqsubseteq$ Professor $\quad$ LecturerOf $\sqsubseteq$ InvolvedWith $\quad$ Assists $\sqsubseteq$ InvolvedWith

$\exists$InvolvedWith$^{-}$ $\sqsubseteq$ Course $\quad$ IntroCourse $\sqsubseteq$ Course $\quad$ 100Series $\sqsubseteq$ IntroCourse

**Query** $q_0 = \exists y \ Professor(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$

**Rewriting of** $q_0$ **w.r.t.** $\mathcal{T}$: ($\sim$ all ways of satisfying $q$)
Example: query rewriting

**TBox \( T \):**

\[\exists \text{LecturerOf} \subseteq \text{Professor} \quad \exists \text{LecturerOf} \subseteq \text{InvolvedWith} \quad \exists \text{Assists} \subseteq \text{InvolvedWith} \]
\[\exists \text{InvolvedWith} \subseteq \text{Course} \quad \exists \text{IntroCourse} \subseteq \text{Course} \quad \exists \text{100Series} \subseteq \text{IntroCourse} \]

**Query** \( q_0 = \exists y \ \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y) \)

**Rewriting of** \( q_0 \) **w.r.t.** \( T \): (~ all ways of satisfying \( q \))

\[\exists y \ \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y) \quad \lor \]

---
Example: query rewriting

TBox $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor}$ \quad $\exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith}$ \quad $\exists \text{Assists} \sqsubseteq \text{InvolvedWith}$

$\exists \text{InvolvedWith}^{-} \sqsubseteq \text{Course}$ \quad $\exists \text{IntroCourse} \sqsubseteq \text{Course}$ \quad $\exists \text{100Series} \sqsubseteq \text{IntroCourse}$

Query $q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$

Rewriting of $q_0$ w.r.t. $\mathcal{T}$: (~ all ways of satisfying $q$)

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$ \quad $\lor$

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y)$ \quad $\lor$
Example: query rewriting

TBox $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor}$  $\text{LecturerOf} \sqsubseteq \text{InvolvedWith}$  $\text{Assists} \sqsubseteq \text{InvolvedWith}$

$\exists \text{InvolvedWith}^- \sqsubseteq \text{Course}$  $\text{IntroCourse} \sqsubseteq \text{Course}$  $\text{100Series} \sqsubseteq \text{IntroCourse}$

Query $q_0 = \exists y \, \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$

Rewriting of $q_0$ w.r.t. $\mathcal{T}$: ($\sim$ all ways of satisfying $q$)

$\exists y \, \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$  $\checkmark$

$\exists y \, \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y)$  $\checkmark$

$\exists y, z \, \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$  $\checkmark$
Example: query rewriting

**TBox** $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor}$ \hspace{1cm} $\exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith}$ \hspace{1cm} $\text{Assists} \sqsubseteq \text{InvolvedWith}$

$\exists \text{InvolvedWith} \neg \sqsubseteq \text{Course}$ \hspace{1cm} $\text{IntroCourse} \sqsubseteq \text{Course}$ \hspace{1cm} $100\text{Series} \sqsubseteq \text{IntroCourse}$

**Query** $q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$

**Rewriting of** $q_0$ **w.r.t.** $\mathcal{T}$: (~ all ways of satisfying $q$)

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$ \hspace{1cm} $\checkmark$

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land 100\text{Series}(y)$ \hspace{1cm} $\checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$ \hspace{1cm} $\checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land 100\text{Series}(y)$ \hspace{1cm} $\checkmark$
Example: query rewriting

**TBox** $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor} \quad \exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith} \quad \exists \text{Assists} \sqsubseteq \text{InvolvedWith}$

$\exists \text{InvolvedWith}^- \sqsubseteq \text{Course} \quad \exists \text{IntroCourse} \sqsubseteq \text{Course} \quad \exists \text{100Series} \sqsubseteq \text{IntroCourse}$

**Query** $q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$

**Rewriting of** $q_0$ **w.r.t.** $\mathcal{T}$: ($\sim$ all ways of satisfying $q$)

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y) \quad \checkmark$

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y) \quad \checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y) \quad \checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y) \quad \checkmark$

$\exists y \text{LecturerOf}(x, y) \land \text{IntroCourse}(y) \quad \checkmark$
Example: query rewriting

**TBox \( \mathcal{T} \):**

\[ \exists \text{LecturerOf} \sqsubseteq \text{Professor} \quad \exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith} \quad \exists \text{Assists} \sqsubseteq \text{InvolvedWith} \]

\[ \exists \text{InvolvedWith} \sqsubseteq \text{Course} \quad \exists \text{IntroCourse} \sqsubseteq \text{Course} \quad \exists \text{100Series} \sqsubseteq \text{IntroCourse} \]

**Query** \( q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y) \)

**Rewriting of** \( q_0 \) **w.r.t.** \( \mathcal{T} \): (\( \sim \) all ways of satisfying \( q \))

\[ \exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y) \quad \checkmark \]

\[ \exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y) \quad \checkmark \]

\[ \exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y) \quad \checkmark \]

\[ \exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y) \quad \checkmark \]

\[ \exists y \text{LecturerOf}(x, y) \land \text{IntroCourse}(y) \quad \checkmark \]

\[ \exists y \text{LecturerOf}(x, y) \land \text{100Series}(y) \quad \checkmark \]
Example: query rewriting

TBox $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor}$  \hspace{1em} $\text{LecturerOf} \sqsubseteq \text{InvolvedWith}$  \hspace{1em} $\text{Assists} \sqsubseteq \text{InvolvedWith}$

$\exists \text{InvolvedWith} \sqsubseteq \text{Course}$  \hspace{1em} $\text{IntroCourse} \sqsubseteq \text{Course}$  \hspace{1em} $100Series \sqsubseteq \text{IntroCourse}$

Query $q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$

Rewriting of $q_0$ w.r.t. $\mathcal{T}$: ($\sim$ all ways of satisfying $q$)

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$  \hspace{1em} $\checkmark$

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land 100Series(y)$  \hspace{1em} $\checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$  \hspace{1em} $\checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land 100Series(y)$  \hspace{1em} $\checkmark$

$\exists y \text{LecturerOf}(x, y) \land \text{IntroCourse}(y)$  \hspace{1em} $\checkmark$

$\exists y \text{LecturerOf}(x, y) \land 100Series(y)$  \hspace{1em} $\checkmark$

$\exists y \text{Professor}(x) \land \text{Assists}(x, y) \land \text{IntroCourse}(y)$  \hspace{1em} $\checkmark$
Example: query rewriting

TBox $\mathcal{T}$:

$\exists \text{LecturerOf} \sqsubseteq \text{Professor}$ \quad $\exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith}$ \quad $\exists \text{Assists} \sqsubseteq \text{InvolvedWith}$

$\exists \text{InvolvedWith} \sqsubseteq \text{Course}$ \quad $\exists \text{IntroCourse} \sqsubseteq \text{Course}$ \quad $\exists 100\text{Series} \sqsubseteq \text{IntroCourse}$

Query $q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)$

Rewriting of $q_0$ w.r.t. $\mathcal{T}$: ($\sim$ all ways of satisfying $q$)

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$ \quad $\checkmark$

$\exists y \text{Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y)$ \quad $\checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y)$ \quad $\checkmark$

$\exists y, z \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y)$ \quad $\checkmark$

$\exists y \text{LecturerOf}(x, y) \land \text{IntroCourse}(y)$ \quad $\checkmark$

$\exists y \text{LecturerOf}(x, y) \land \text{100Series}(y)$ \quad $\checkmark$

$\exists y \text{Professor}(x) \land \text{Assists}(x, y) \land \text{IntroCourse}(y)$ \quad $\checkmark$

$\exists y \text{Professor}(x) \land \text{Assists}(x, y) \land \text{100Series}(y)$ \quad $\checkmark$
Example: query rewriting

**TBox \( T \):**

\[ \exists \text{LecturerOf} \sqsubseteq \text{Professor} \quad \exists \text{LecturerOf} \sqsubseteq \exists \text{InvolvedWith} \quad \exists \text{Assists} \sqsubseteq \exists \text{InvolvedWith} \]

\[ \exists \text{InvolvedWith} \sqsubseteq \exists \text{Course} \quad \exists \text{IntroCourse} \sqsubseteq \exists \text{Course} \quad \exists \text{100Series} \sqsubseteq \exists \text{IntroCourse} \]

**Query** \( q_0 = \exists y \text{Professor}(x) \land \exists y \text{InvolvedWith}(x, y) \land \exists y \text{IntroCourse}(y) \)

**Rewriting of** \( q_0 \) **w.r.t.** \( T \): (\( \sim \) all ways of satisfying \( q \))

\[ \exists y \text{Professor}(x) \land \exists y \text{InvolvedIn}(x, y) \land \exists y \text{IntroCourse}(y) \quad \checkmark \]

\[ \exists y \text{Professor}(x) \land \exists y \text{InvolvedIn}(x, y) \land \exists y \text{100Series}(y) \quad \checkmark \]

\[ \exists y, z \text{LecturerOf}(x, z) \land \exists y \text{InvolvedIn}(x, y) \land \exists y \text{IntroCourse}(y) \quad \checkmark \]

\[ \exists y, z \text{LecturerOf}(x, z) \land \exists y \text{InvolvedIn}(x, y) \land \exists y \text{100Series}(y) \quad \checkmark \]

\[ \exists y \text{LecturerOf}(x, y) \land \exists y \text{IntroCourse}(y) \quad \checkmark \]

\[ \exists y \text{LecturerOf}(x, y) \land \exists y \text{100Series}(y) \quad \checkmark \]

\[ \exists y \text{Professor}(x) \land \exists y \text{Assists}(x, y) \land \exists y \text{IntroCourse}(y) \quad \checkmark \]

\[ \exists y \text{Professor}(x) \land \exists y \text{Assists}(x, y) \land \exists y \text{100Series}(y) \quad \checkmark \]

\[ \exists y, z \text{LecturerOf}(x, z) \land \exists y \text{Assists}(x, y) \land \exists y \text{IntroCourse}(y) \quad \checkmark \]
Example: query rewriting

**TBox \(T\):**

\[\exists \text{LecturerOf} \sqsubseteq \text{Professor} \quad \exists \text{LecturerOf} \sqsubseteq \text{InvolvedWith} \quad \exists \text{Assists} \sqsubseteq \text{InvolvedWith}\]

\[\exists \text{InvolvedWith} \sqsubseteq \text{Course} \quad \exists \text{IntroCourse} \sqsubseteq \text{Course} \quad \exists \text{100Series} \sqsubseteq \text{IntroCourse}\]

**Query \(q_0 = \exists y \text{Professor}(x) \land \text{InvolvedWith}(x, y) \land \text{IntroCourse}(y)\)**

**Rewriting of \(q_0\) w.r.t. \(T\):** (\(~\) all ways of satisfying \(q\))

\[
\begin{align*}
\exists y & \text{ Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y) & \checkmark \\
\exists y & \text{ Professor}(x) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y) & \checkmark \\
\exists y, z & \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{IntroCourse}(y) & \checkmark \\
\exists y, z & \text{LecturerOf}(x, z) \land \text{InvolvedIn}(x, y) \land \text{100Series}(y) & \checkmark \\
& \quad \text{Lecturer}(x, y) \land \text{IntroCourse}(y) & \checkmark \\
& \quad \text{LecturerOf}(x, y) \land \text{100Series}(y) & \checkmark \\
\exists y & \text{ Professor}(x) \land \text{Assists}(x, y) \land \text{IntroCourse}(y) & \checkmark \\
\exists y & \text{ Professor}(x) \land \text{Assists}(x, y) \land \text{100Series}(y) & \checkmark \\
\exists y, z & \text{LecturerOf}(x, z) \land \text{Assists}(x, y) \land \text{IntroCourse}(y) & \checkmark \\
\exists y, z & \text{LecturerOf}(x, z) \land \text{Assists}(x, y) \land \text{100Series}(y) & \checkmark \\
\end{align*}
\]
Example: evaluation of rewritten query

Now suppose that our ABox $\mathcal{A}$ contains the following facts:

\begin{align*}
  & \text{Professor}(jane) \quad \text{Assists}(jane, \text{PSY300}) \quad \text{LecturerOf}(\text{maria}, \text{INF101}) \\
  & \text{IntroCourse}(\text{PSY300}) \quad 100\text{Series}(\text{INF101}) \quad \text{Assists}(\text{paul}, \text{INF101})
\end{align*}

To compute the certain answers to $q_0$ w.r.t. $(\mathcal{T}, \mathcal{A})$, we evaluate the rewritten query over $\mathcal{I}_\mathcal{A}$. 
Now suppose that our ABox $\mathcal{A}$ contains the following facts:

- $Professor(jane)$
- $Assists(jane, PSY300)$
- $LecturerOf(maria, INF101)$
- $IntroCourse(PSY300)$
- $100Series(INF101)$
- $Assists(paul, INF101)$

To compute the certain answers to $q_0$ w.r.t. $(\mathcal{T}, \mathcal{A})$, we evaluate the rewritten query over $\mathcal{I}_\mathcal{A}$.

**Result:** $jane$ (via 7th disjunct), $maria$ (via 6th disjunct)
Now suppose that our ABox $\mathcal{A}$ contains the following facts:

\[
\begin{align*}
\text{Professor}(\text{jane}) & \quad \text{Assists}(\text{jane}, \text{PSY300}) & \quad \text{LecturerOf}(\text{maria}, \text{INF101}) \\
\text{IntroCourse}(\text{PSY300}) & \quad \text{100Series}(\text{INF101}) & \quad \text{Assists}(\text{paul}, \text{INF101})
\end{align*}
\]

To compute the certain answers to $q_0$ w.r.t. $(\mathcal{T}, \mathcal{A})$, we evaluate the rewritten query over $\mathcal{I}_\mathcal{A}$.

**Result**: $\text{jane}$ (via 7th disjunct), $\text{maria}$ (via 6th disjunct)

**Certain answers to $q$ w.r.t. $(\mathcal{T}, \mathcal{A})$**: $\text{jane, maria}$
Example: evaluation of rewritten query

Now suppose that our ABox $\mathcal{A}$ contains the following facts:

- $\text{Professor}(\text{jane})$
- $\text{Assists}(\text{jane}, \text{PSY300})$
- $\text{LecturerOf}(\text{maria}, \text{INF101})$
- $\text{IntroCourse}(\text{PSY300})$
- $\text{100Series}(\text{INF101})$
- $\text{Assists}(\text{paul}, \text{INF101})$

To compute the certain answers to $q_0$ w.r.t. $(\mathcal{T}, \mathcal{A})$, we evaluate the rewritten query over $\mathcal{I}_{\mathcal{A}}$.

**Result:** $\text{jane}$ (via 7th disjunct), $\text{maria}$ (via 6th disjunct)

**Certain answers to $q$ w.r.t. $(\mathcal{T}, \mathcal{A})$:** $\text{jane}$, $\text{maria}$

Now we show how to compute a rewriting.
Applicable axioms

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 $P(x_1, x_2)$ if:

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

Let $I$ be an inclusion that is applicable to atom $\alpha$.

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

- if $\alpha = A(x)$ and $I = B \sqsubseteq A$, then $\text{ra}(\alpha, I) = B(x)$
- if $\alpha = A(x)$ and $I = \exists R \sqsubseteq A$, then $\text{ra}(\alpha, I) = R(x, \_)$
- if $\alpha = A(x)$ and $I = \exists R^- \sqsubseteq A$, then $\text{ra}(\alpha, I) = R(\_, x)$
- if $\alpha = R(x, \_)$ and $I = A \sqsubseteq \exists R$, then $\text{ra}(\alpha, I) = A(x)$
- if $\alpha = R(x, \_)$ and $I = \exists S \sqsubseteq \exists R$, then $\text{ra}(\alpha, I) = S(x, \_)$
- if $\alpha = R(x, \_)$ and $I = \exists S^- \sqsubseteq \exists R$, then $\text{ra}(\alpha, I) = S(\_, x)$
- if $\alpha = R(\_, x)$ and $I = A \sqsubseteq \exists R^-$, then $\text{ra}(\alpha, I) = A(x)$
- if $\alpha = R(\_, x)$ and $I = \exists S \sqsubseteq \exists R^-$, then $\text{ra}(\alpha, I) = S(x, \_)$
- if $\alpha = R(\_, x)$ and $I = \exists S^- \sqsubseteq \exists R^-$, then $\text{ra}(\alpha, I) = S(\_, x)$
- if $\alpha = R(x, y)$ and $I = S \sqsubseteq R$ or $I = S^- \sqsubseteq R^-$, then $\text{ra}(\alpha, I) = S(x, y)$
- if $\alpha = R(x, y)$ and $I = S \sqsubseteq R^-$ or $I = S^- \sqsubseteq R$, then $\text{ra}(\alpha, I) = S(y, x)$
Rewriting algorithm PerfectRef

**Input:** conjunctive query $q_0$, TBox $T$

**Output:** set of CQs whose disjunction is a rewriting of $q_0$ given $T$

$PR := \{\tau(q_0)\}$

**repeat** until $PR' = PR$

$PR' := PR$

**for each** $q \in PR'$ that has not yet been considered **do**

**for each** $\alpha \in q$ and $I \in T$ **do**

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

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

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

if $\alpha$ and $\beta$ unify

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

**return** $PR$

$\tau$: replaces existential variables that occur only once by ‘_’

reduce: replaces $\alpha$ and $\beta$ by most general unifier
Example: PerfectRef algorithm

Suppose

\[ \mathcal{T} = \{ (1) \; R \subseteq S \; (2) \; A \subseteq \exists S^- \; (3) \; B \subseteq A \} \]

\[ q_0 = \exists x \; S(x, y) \]
Suppose

\[ T = \{ (1) \ R \sqsubseteq S \quad (2) \ A \sqsubseteq \exists S^- \quad (3) \ B \sqsubseteq A \} \]

\[ q_0 = \exists x \ S(x, y) \]

Initially, \( PR = \{ \tau(q_0) \} = \{ S(\_, y) \} \).
Example: PerfectRef algorithm

Suppose

\[ \mathcal{T} = \{ (1) R \subseteq S \ (2) A \subseteq \exists S^- \ (3) B \subseteq A \} \]
\[ q_0 = \exists x \ S(x, y) \]

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

First iteration of PerfectRef adds the following queries:

\[ q_1 = R(_, y) \quad \text{apply (1) to the only atom of } \tau(q_0) \]
\[ q_2 = A(y) \quad \text{apply (2) to the only atom of } \tau(q_0) \]
Example: PerfectRef algorithm

Suppose

\[ \mathcal{T} = \{ (1) \ R \sqsubseteq S \ (2) \ A \sqsubseteq \exists S^- \ (3) \ B \sqsubseteq A \} \]
\[ q_0 = \exists x \ S(x, y) \]

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

First iteration of PerfectRef adds the following queries:

\[ q_1 = R(\_, y) \quad \text{apply (1) to the only atom of } \tau(q_0) \]
\[ q_2 = A(y) \quad \text{apply (2) to the only atom of } \tau(q_0) \]

In second iteration, we add

\[ q_3 = B(y) \quad \text{apply (3) to the only atom of } q_2 \]

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

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

query:
$q_0 = \text{Prof}(x) \land \text{InvWith}(x, y) \land \text{IntroC}(y)$ (note: $\tau(q_0) = q_0$)
Example: PerfectRef algorithm

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

query:  
$q_0 = Prof(x) \land InvWith(x, y) \land IntroC(y)$  
(note: $\tau(q_0) = q_0$)

First iteration of PerfectRef adds the following queries:

$q_1 = LectOf(x, _) \land InvWith(x, y) \land IntroC(y)$

$q_2 = Prof(x) \land LectOf(x, y) \land IntroC(y)$

$q_3 = Prof(x) \land InvWith(x, y) \land 100S(y)$
Example: PerfectRef algorithm

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

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

First iteration of PerfectRef adds the following queries:

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

Second iteration adds:

$q_4 = \text{LectOf}(x, \_ ) \land \text{LectOf}(x, y) \land \text{IntroC}(y)$
$q_5 = \text{LectOf}(x, \_ ) \land \text{InvWith}(x, y) \land 100S(y)$
$q_6 = \text{Prof}(x) \land \text{LectOf}(x, y) \land 100S(y)$
Example: PerfectRef algorithm

TBox: 
1. \( \exists \text{LectOf} \sqsubseteq \text{Prof} \)
2. \( \text{LectOf} \sqsubseteq \text{InvWith} \)
3. \( 100S \sqsubseteq \text{IntroC} \)

query: 
\( q_0 = \text{Prof}(x) \land \text{InvWith}(x, y) \land \text{IntroC}(y) \) (note: \( \tau(q_0) = q_0 \))

First iteration of PerfectRef adds the following queries:

\( q_1 = \text{LectOf}(x, \_ ) \land \text{InvWith}(x, y) \land \text{IntroC}(y) \)
\( q_2 = \text{Prof}(x) \land \text{LectOf}(x, y) \land \text{IntroC}(y) \)
\( q_3 = \text{Prof}(x) \land \text{InvWith}(x, y) \land 100S(y) \)

Second iteration adds:

\( q_4 = \text{LectOf}(x, \_ ) \land \text{LectOf}(x, y) \land \text{IntroC}(y) \)
\( q_5 = \text{LectOf}(x, \_ ) \land \text{InvWith}(x, y) \land 100S(y) \)
\( q_6 = \text{Prof}(x) \land \text{LectOf}(x, y) \land 100S(y) \)

Third iteration gives:

\( q_7 = \text{LectOf}(x, \_ ) \land \text{LectOf}(x, y) \land 100S(y) \)
\( q_8 = \text{LectOf}(x, y) \land \text{IntroC}(y) \) (unifying atoms in \( q_4 \))
Example: PerfectRef algorithm

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

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

First iteration of PerfectRef adds the following queries:

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

Second iteration adds:

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

Third iteration gives:

$q_7 = \text{LectOf}(x, \_ ) \land \text{LectOf}(x, y) \land 100S(y)$
$q_8 = \text{LectOf}(x, y) \land \text{IntroC}(y)$  
(unifying atoms in $q_4$)

Fourth and final iteration yields:

$q_9 = \text{LectOf}(x, y) \land 100S(y)$  
(unifying atoms in $q_7$)
Correctness of rewriting algorithm

The following result shows the correctness of PerfectRef:

**Theorem.** Let \( q \) be a conjunctive query, \((\mathcal{T}, \mathcal{A})\) be a satisfiable DL-Lite\(_R\) KB, and \( \bar{a} \) a tuple of individuals from \( \mathcal{A} \). Then

\[
\bar{a} \in \text{cert}(q, (\mathcal{T}, \mathcal{A})) \iff \mathcal{I}_\mathcal{A} \models \Psi[\bar{a}]
\]

where \( \Psi \) is the *disjunction of queries in PerfectRef\((q, \mathcal{T})\) (with each \_ replaced by a fresh existentially quantified variable).
Satisfiability via query rewriting

Our query rewriting approach only works if the input KB is satisfiable. Therefore, we 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 $DL-Lite_\mathcal{R}$ 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 $unsat(B \sqsubseteq \neg C)$ the CQ that describes when $B \sqsubseteq \neg C$ is not satisfied.

**Example:**

- $unsat(A \sqsubseteq \neg D) = \exists x \ A(x) \land D(x)$
- $unsat(\exists R \sqsubseteq \neg \exists S^-) = \exists x, y, z \ R(x, y) \land S(z, x)$
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.

**Example:**

▶ $\text{unsat}(A \sqsubseteq \neg D) = \exists x \ A(x) \land D(x)$
▶ $\text{unsat}(\exists R \sqsubseteq \neg \exists S^-) = \exists x, y, z \ R(x, y) \land S(z, x)$
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.

Example:

- $\text{unsat}(A \sqsubseteq \neg D) = \exists x \ A(x) \land D(x)$
- $\text{unsat}(\exists R \sqsubseteq \neg \exists S^-) = \exists x, y, z \ R(x, y) \land S(z, x)$

Evaluate the following Boolean union of conjunctive queries

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

over $\mathcal{I}_A$.

Evaluation returns yes $\iff (T, A)$ is unsatisfiable
Complexity of reasoning in *DL-Lite*

By complexity of conjunctive query answering, we mean the complexity of deciding if $\bar{a} \in \text{cert}(q, (\mathcal{T}, \mathcal{A}))$. 

Two complexity measures:
- **combined complexity**: in terms of the size of KB and query data complexity
- **data complexity**: only in terms of the size of the ABox

Appropriate when $|\mathcal{A}|$ much bigger than $|\mathcal{T}|$ and $|q|$ (often the case)

Since the query rewriting step is independent of the data, and query evaluation over databases has data complexity of $\text{AC}^0$, we obtain:

**Theorem.** For *DL-Lite* $\mathcal{R}$, satisfiability and CQ answering are in $\text{AC}^0$ for data complexity.

Note: $\text{AC}^0 \subset \text{LOGSPACE} \subset \text{NLOGSPACE} \subset \text{PTIME}$
Complexity of reasoning in $DL$-$Lite$

By complexity of conjunctive query answering, we mean the complexity of deciding if $\bar{a} \in \text{cert}(q, (T, A))$.

Two complexity measures:

▶ combined complexity: in terms of the size of KB and query
▶ data complexity: only in terms of the size of the ABox
   ▶ appropriate when $|A|$ much bigger than $|T|$ and $|q|$ (often the case)
Complexity of reasoning in *DL-Lite*

By complexity of conjunctive query answering, we mean the complexity of deciding if \( \vec{a} \in \text{cert}(q, (\mathcal{T}, \mathcal{A})) \).

Two complexity measures:

- **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 \(|\mathcal{T}|\) and \(|q|\) (often the case)

Since the query rewriting step is independent of the data, and query evaluation over databases has data complexity of \(AC^0\), we obtain:

**Theorem.** For *DL-Lite*\(_R\), satisfiability and CQ answering are in \(AC^0\) for data complexity.

*Note:* \(AC^0 \subset LOGSPACE \subset NLOGSPACE \subset PTIME\)
Complexity of reasoning in *DL-Lite*

For combined complexity, CQ 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 for combined complexity.

However, satisfiability and instance checking are tractable:

**Theorem.** For *DL-Lite*$_R$, satisfiability and instance checking are NLOGSPACE-complete for combined complexity.
Practical query rewriting algorithms

Adopt more compact formats of rewritings to avoid blowup in size

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 model (which has witnesses for $\exists R$), then filter answers to remove false positives

Example system: Ontop (ontop-vkg.org)