ONTOLOGIES & DESCRIPTION LOGICS

Parcours IA - Représentation des connaissances

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 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

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 $\sim$ select-project-join queries in SQL

USING ONTOLOGIES TO ACCESS DATA

Aim: enrich databases (DBs) with ontologies

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

Desiderata:

- efficiency is crucial – must scale up to huge datasets
- instance queries too simple – want expressive queries like in DBs
  - conjunctive queries $\sim$ 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 $B := A \mid \exists S \qquad S := r \mid r^-$

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

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

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

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

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

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

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

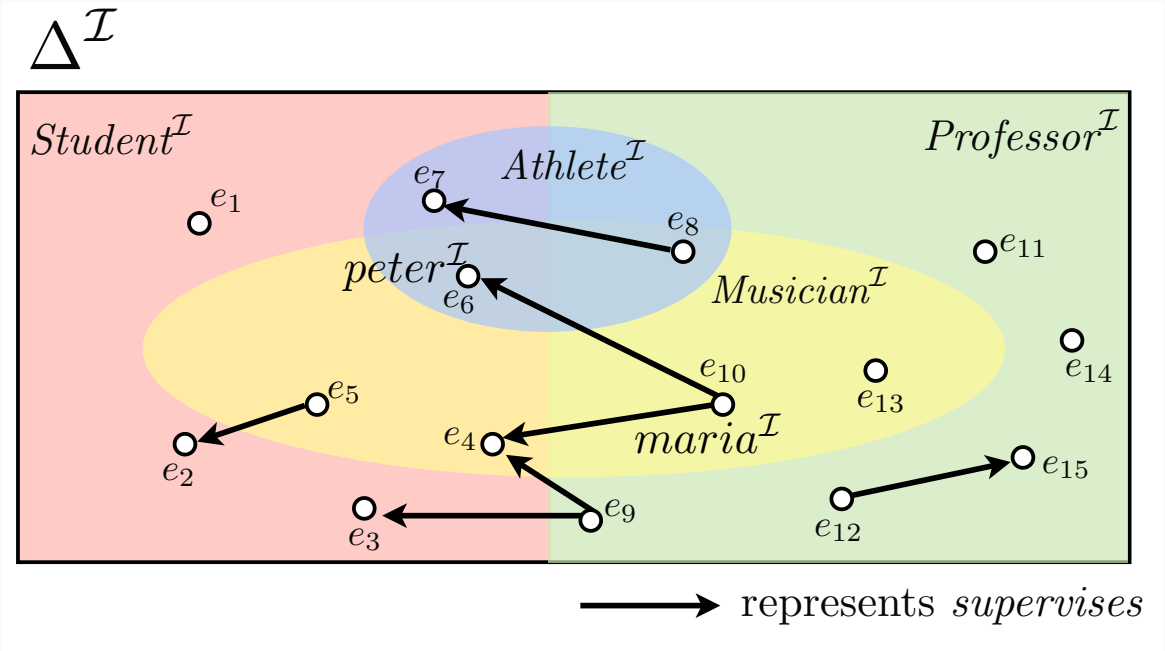
Satisfaction of a Boolean CQ in an interpretation:

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

EXAMPLE: CERTAIN ANSWERS

DL-Lite ontology:

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

ABox:

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

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

EXAMPLE: CERTAIN ANSWERS

DL-Lite ontology:

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

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

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:

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

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

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

EXAMPLE: QUERY REWRITING IN DL-LITE

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

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

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 $\mathcal{T}$:

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

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

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

$$\begin{array}{rcl} q'(x) & = & \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y) \quad \vee \quad \mathsf{Prof}(x) \\ & & \vee \quad \exists y.\mathsf{Researcher}(x) \wedge \mathsf{teaches}(x, y) \end{array}$$

EXAMPLE: QUERY REWRITING IN DL-LITE

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

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

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

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

$$\begin{array}{rcl} q'(x) & = & \exists y.\mathsf{Faculty}(x) \wedge \mathsf{teaches}(x, y) \quad \vee \quad \mathsf{Prof}(x) \\ & & \vee \quad \exists y.\mathsf{Researcher}(x) \wedge \mathsf{teaches}(x, y) \end{array}$$

Evaluating the rewritten query over the earlier dataset

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

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

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

REWRITING ALGORITHM: ATOMS

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

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

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

REWRITING ALGORITHM: ATOMS

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

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

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

REWRITING ALGORITHM: ATOMS

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

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

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

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

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

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

Functions τ and *merge* described on next slide

Function τ:

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

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

Function *merge*:

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

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

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:

$q_1 = r(x, \_) \wedge B(x) \wedge z = x$ *merge* operation followed by τ

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:

$q_1 = r(x, \_) \wedge B(x) \wedge z = x$ *merge* operation followed by τ

In second iteration, we consider q_1 and add

$q_2 = A(x) \wedge B(x) \wedge z = x$ apply $A \sqsubseteq \exists r$ to r-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)

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

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

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

First iteration of PerfectRef adds the following queries:

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

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

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

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

First iteration of PerfectRef adds the following queries:

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

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

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

From preceding queries, we get:

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

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

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

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

First iteration of PerfectRef adds the following queries:

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

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

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

From preceding queries, we get:

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

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

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

Further queries obtained when considering q_4:

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

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

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

First iteration of PerfectRef adds the following queries:

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

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

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

From preceding queries, we get:

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

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

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

Further queries obtained when considering q_4:

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

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

Final iteration yields:

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

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 '_').

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 $rewrite(q, \mathcal{T})$ 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 $q^r = rewrite(q, \mathcal{T})$. Then*
>
> $$\vec{a} \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{A})) \quad \Leftrightarrow \quad \mathcal{I}_{\mathcal{A}} \models q^r(\vec{a})$$

SATISFIABILITY VIA QUERY REWRITING

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

- thus: also need a way to test KB satisfiability

SATISFIABILITY VIA QUERY REWRITING

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

- thus: also need a way to test KB satisfiability

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

SATISFIABILITY VIA QUERY REWRITING

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

- thus: also need a way to test KB satisfiability

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

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

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

SATISFIABILITY VIA QUERY REWRITING

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

- thus: also need a way to test KB satisfiability

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

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

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

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

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

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

Satisfiability and instance checking are tractable:

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

NLOGSPACE $\subseteq$ PTIME

COMPLEXITY OF REASONING IN DL-LITE

Satisfiability and instance checking are tractable:

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

NLOGSPACE $\subseteq$ PTIME

What about ontology-mediated query answering?

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

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

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

DATA COMPLEXITY OF QUERYING IN DL-LITE

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

Distinguish two ways of measuring complexity:

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

Results stated so far: combined complexity measure

DATA COMPLEXITY OF QUERYING IN DL-LITE

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

Distinguish two ways of measuring complexity:

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

Results stated so far: combined complexity measure

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

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

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

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

Adopt more compact formats for 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)